Analysis of Distributed Probabilistic Systems: Limitations and Possibilities

Pedro R. D'Argenio

Universidad Nacional de Córdoba
CONICET

Joint work with Sergio Giro, Luis M. Ferrer Fioriti, Georgel Calin, Pepijn Crouzen, Ernst Moritz Hahn, Lijun Zhang, Silvia Pelozo

19-Jun-2014 - OPCT - Bertinoro

CONICET

UNC

Overview

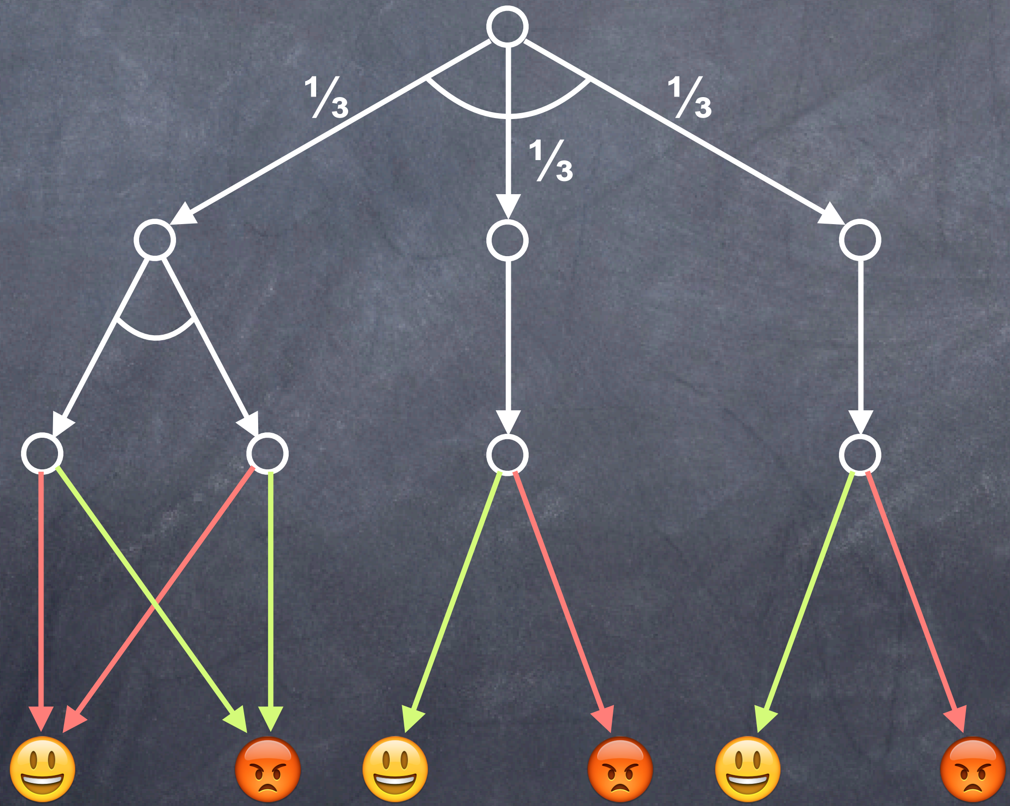
- Motivation
- Distributed Schedulers
- Strongly Distributed Schedulers
- Distributed Schedulers under secrecy
- (Un)decidability results
- Concluding remarks

CONICET

UNC

Model Checking
Probabilistic Concurrent Systems

- Nondeterminism resolved through schedulers

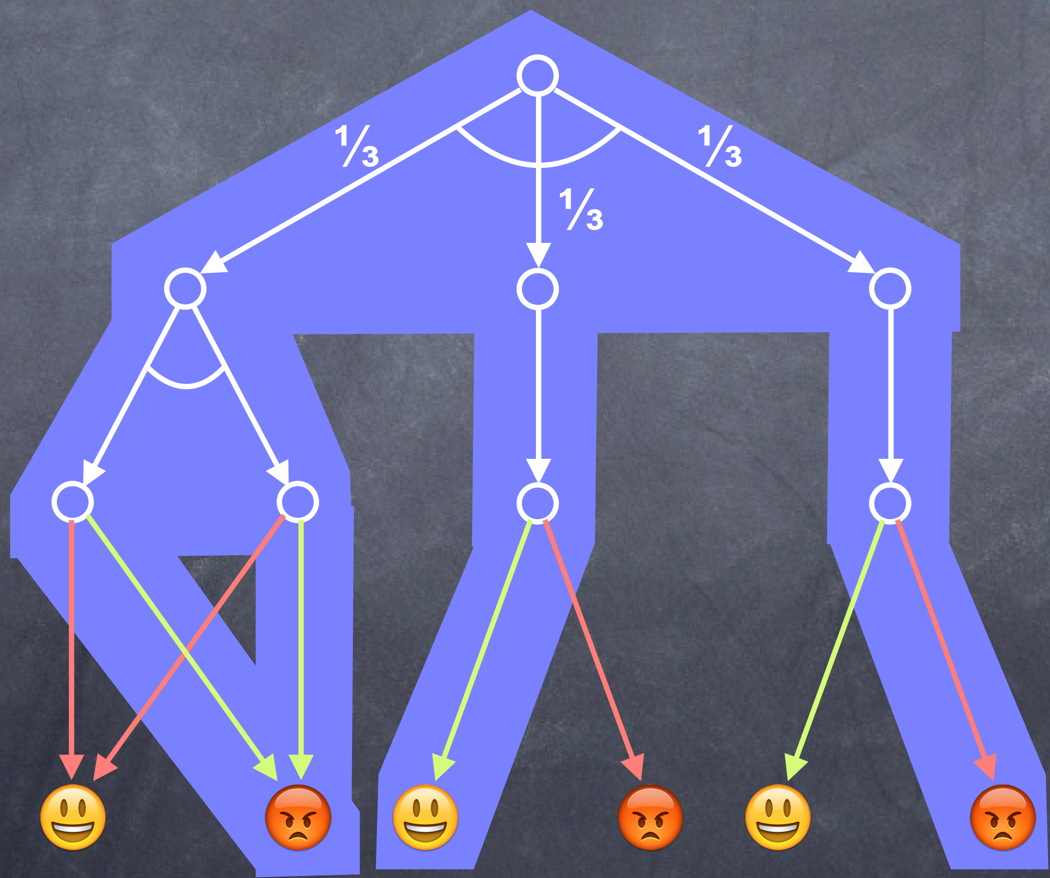


CONICET

UNC

Model Checking Probabilistic Concurrent Systems

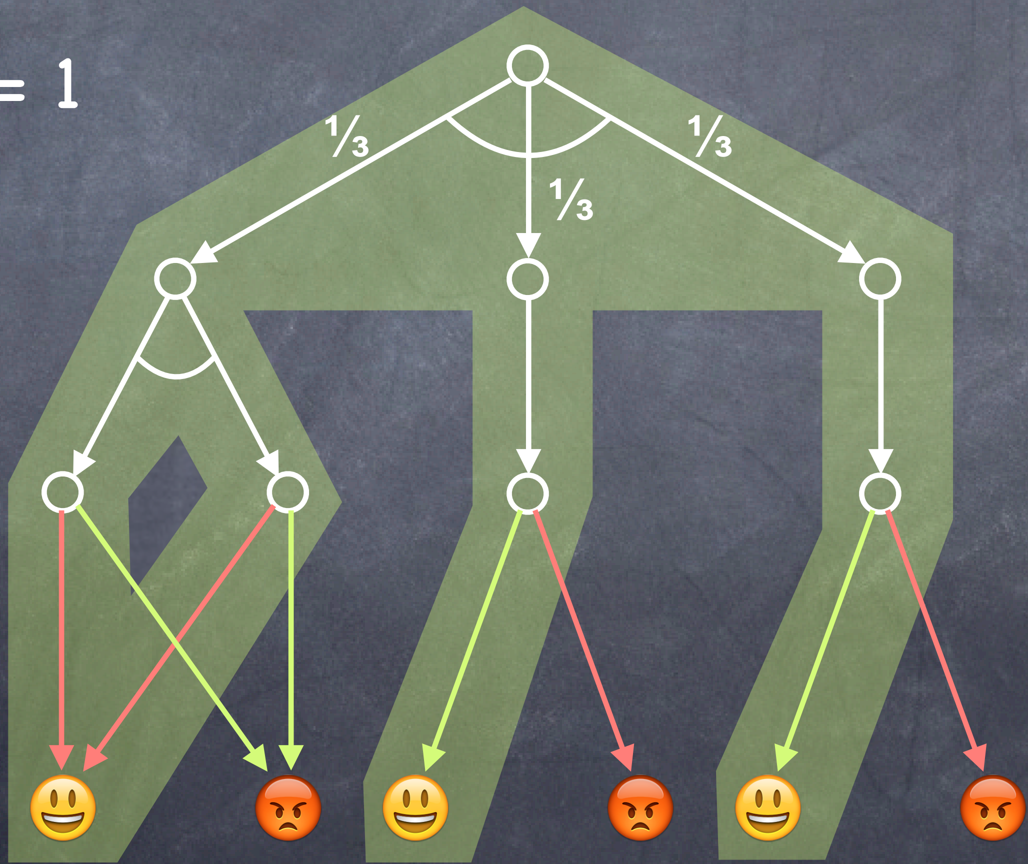
- Nondeterminism resolved through schedulers



Model Checking
Probabilistic Concurrent Systems

- Nondeterminism resolved through schedulers
- Quantifies over all possible schedulers

sup P(F😃) = 1



CONICET

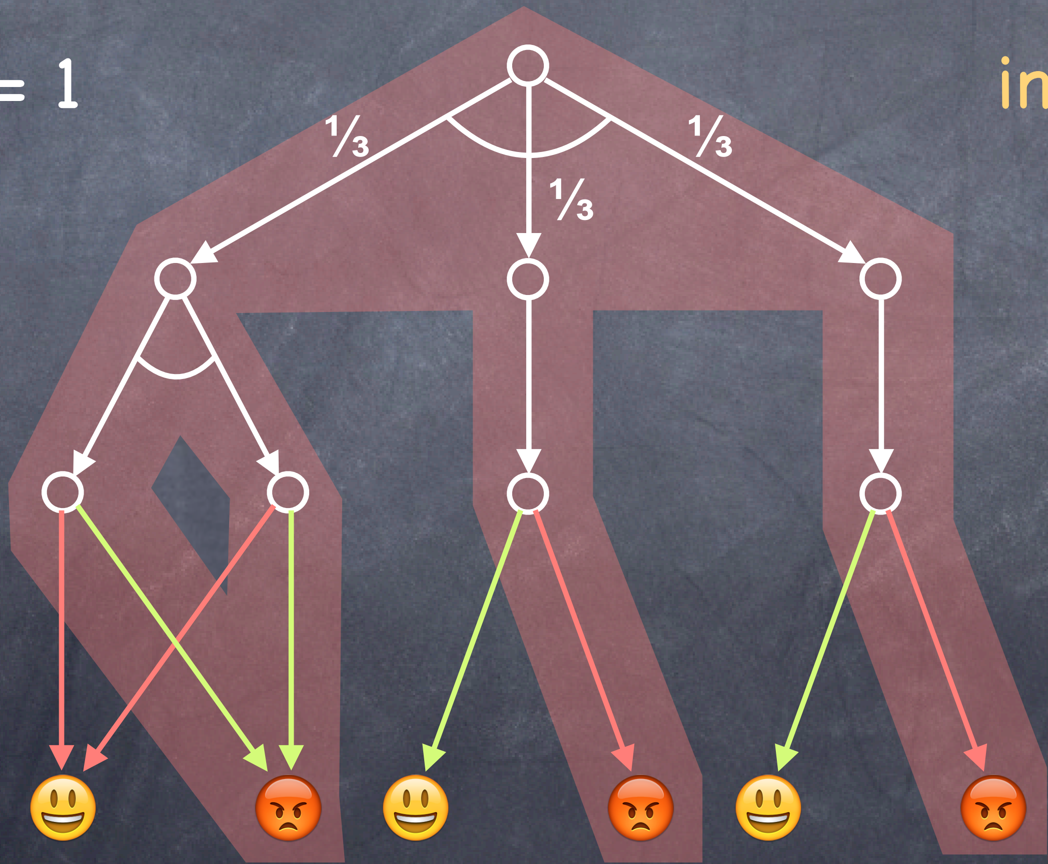
UNC

Model Checking Probabilistic Concurrent Systems

- Nondeterminism resolved through schedulers
- Quantifies over all possible schedulers

sup P(F😃) = 1

inf P(F😃) = 0

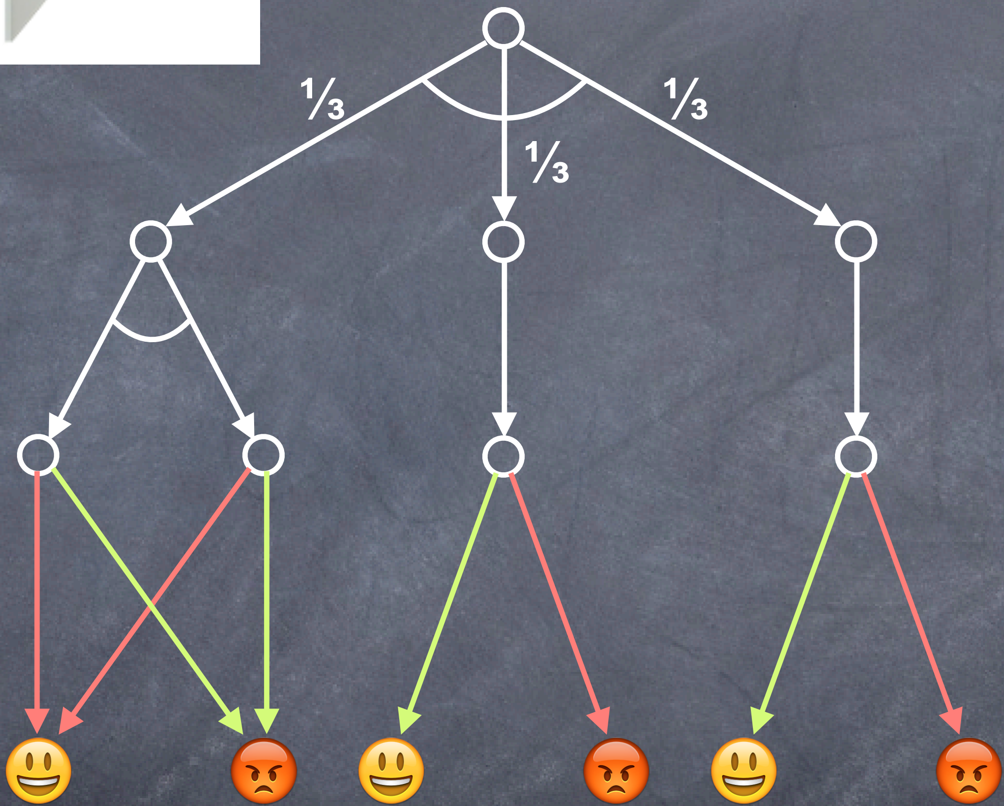


CONICET

UNC

odel Checking
tic Concurrent Systems

Monty Hall problem



choose door

open door

keep door

switch door

sup P(F😃) = 2/3 inf P(F😃) = 1/3

CONICET

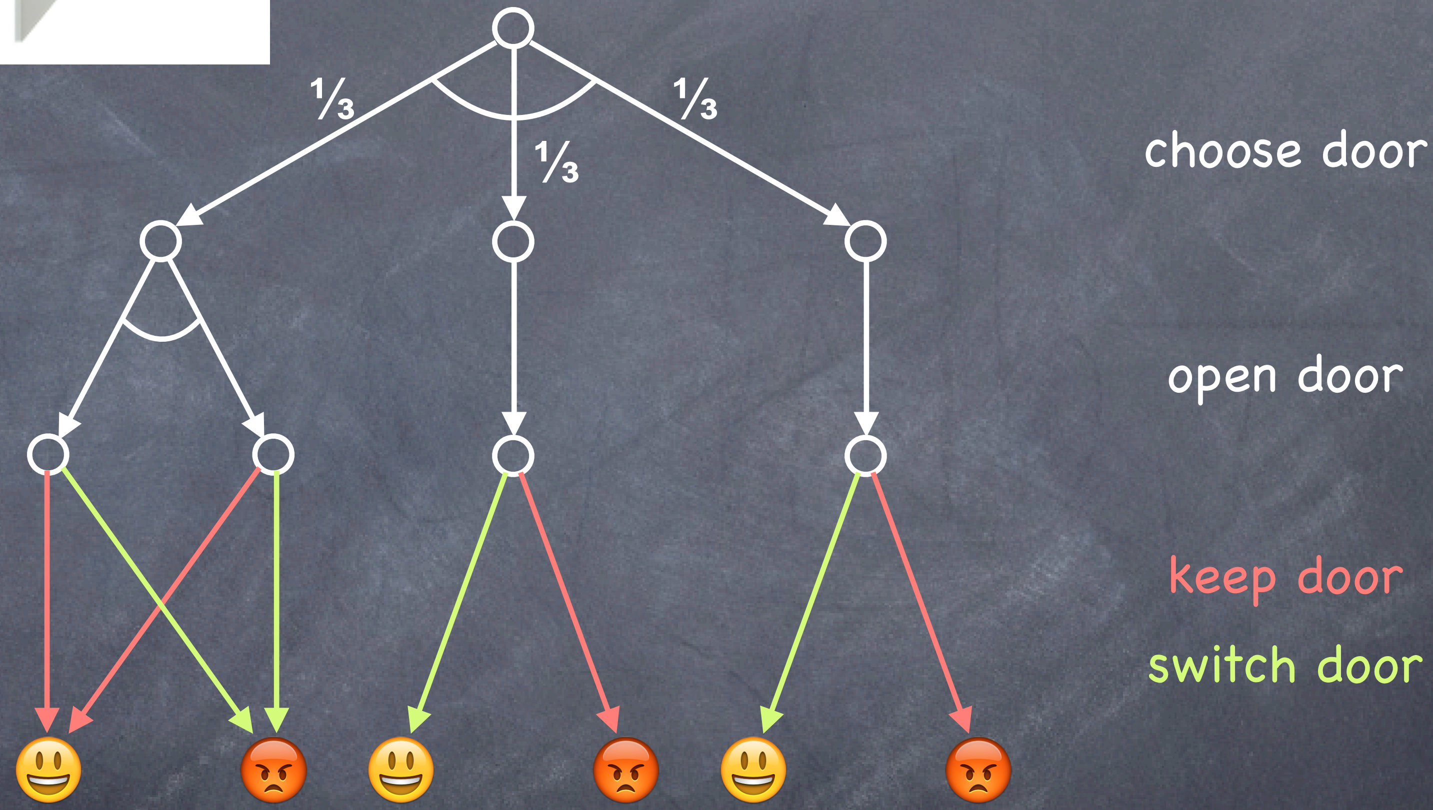
UNC

Probabilistic model checking provides a safe over-approximation of the actual probability value

…l Checking …
… Concurrent S…

All schedulers are too many!

Monty Hall problem

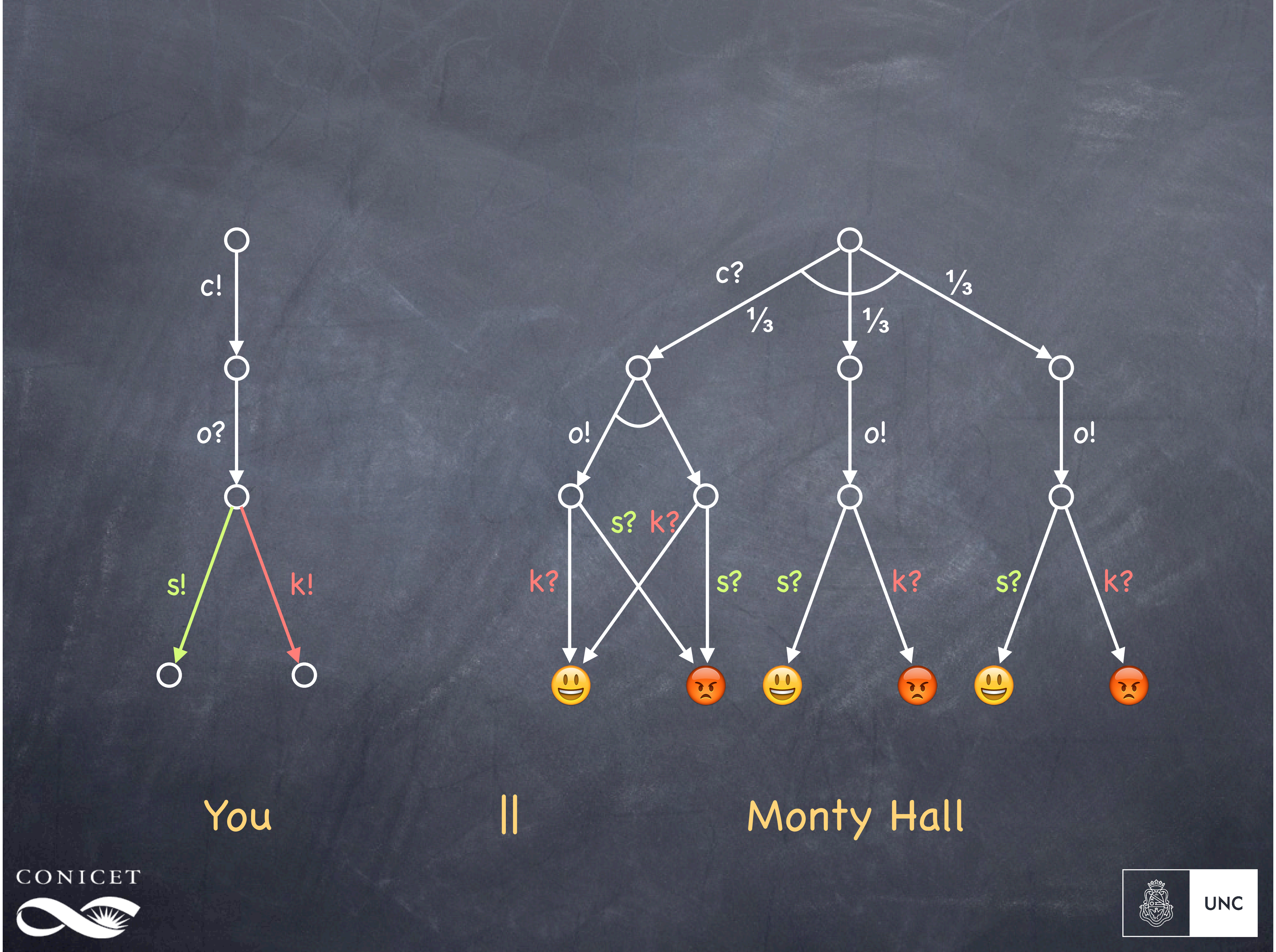


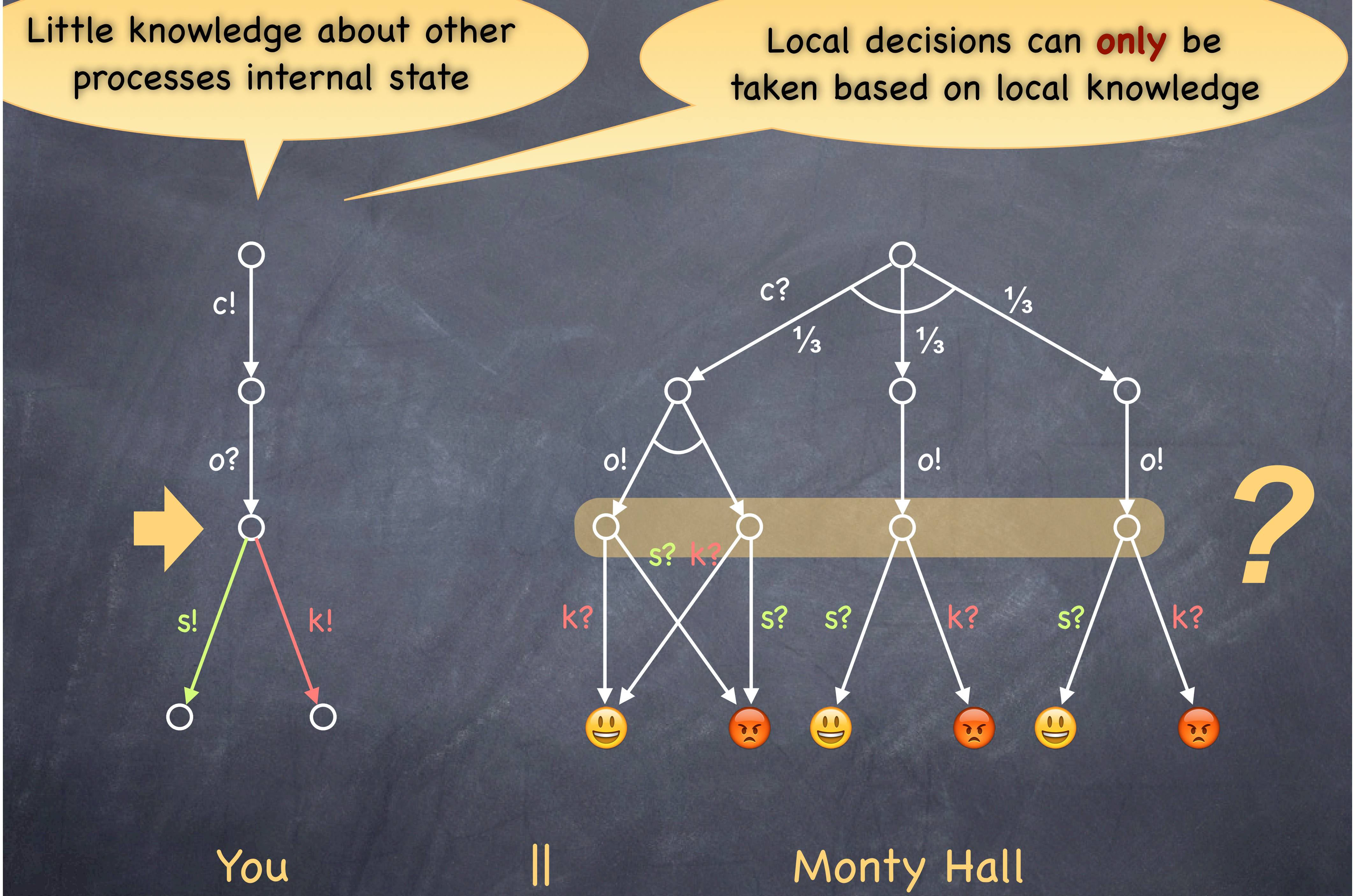
sup P(F😃) = 2/3

inf P(F😃) = 1/3

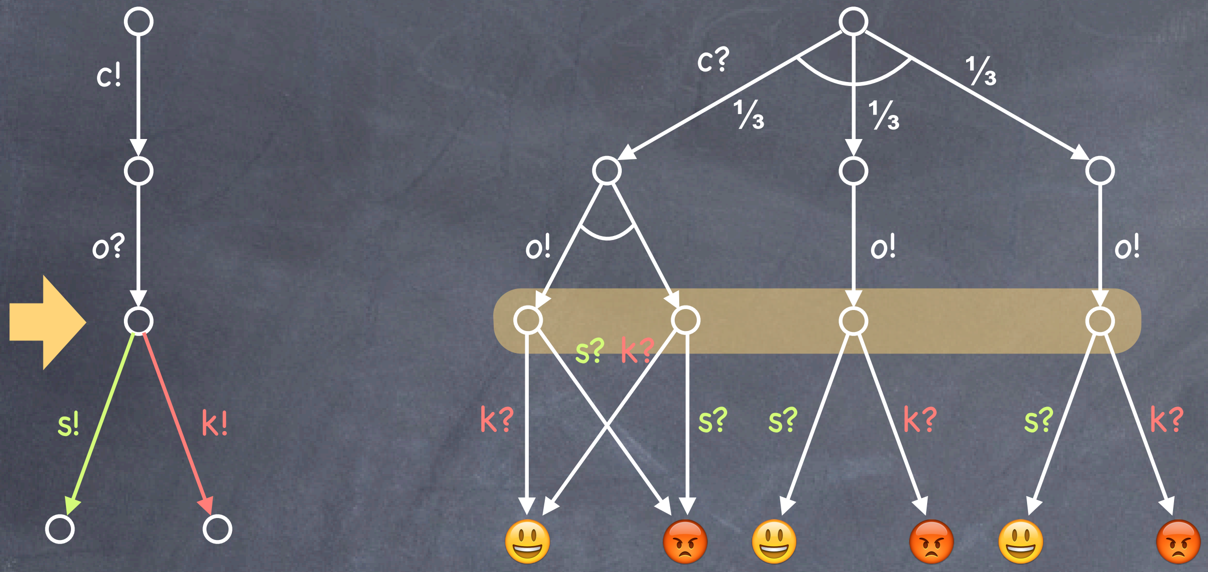
CONICET

UNC

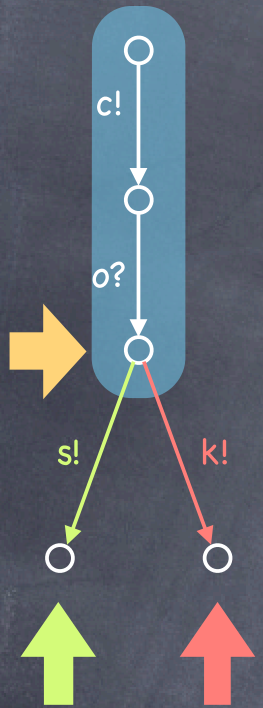

c!
o?
s!
k!
c?
1/3
1/3
1/3
o!
o!
o!
s? k?
k?
s?
s?
k?
s?
k?
You
||
Monty Hall
CONICET
UNC

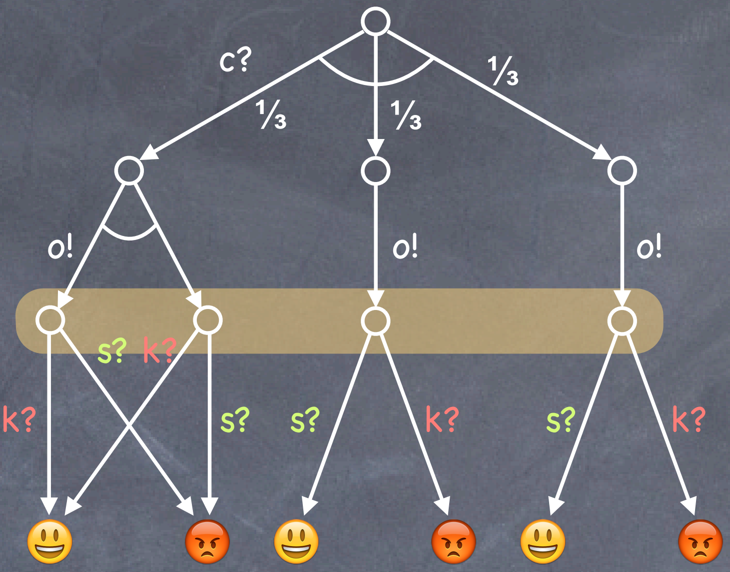


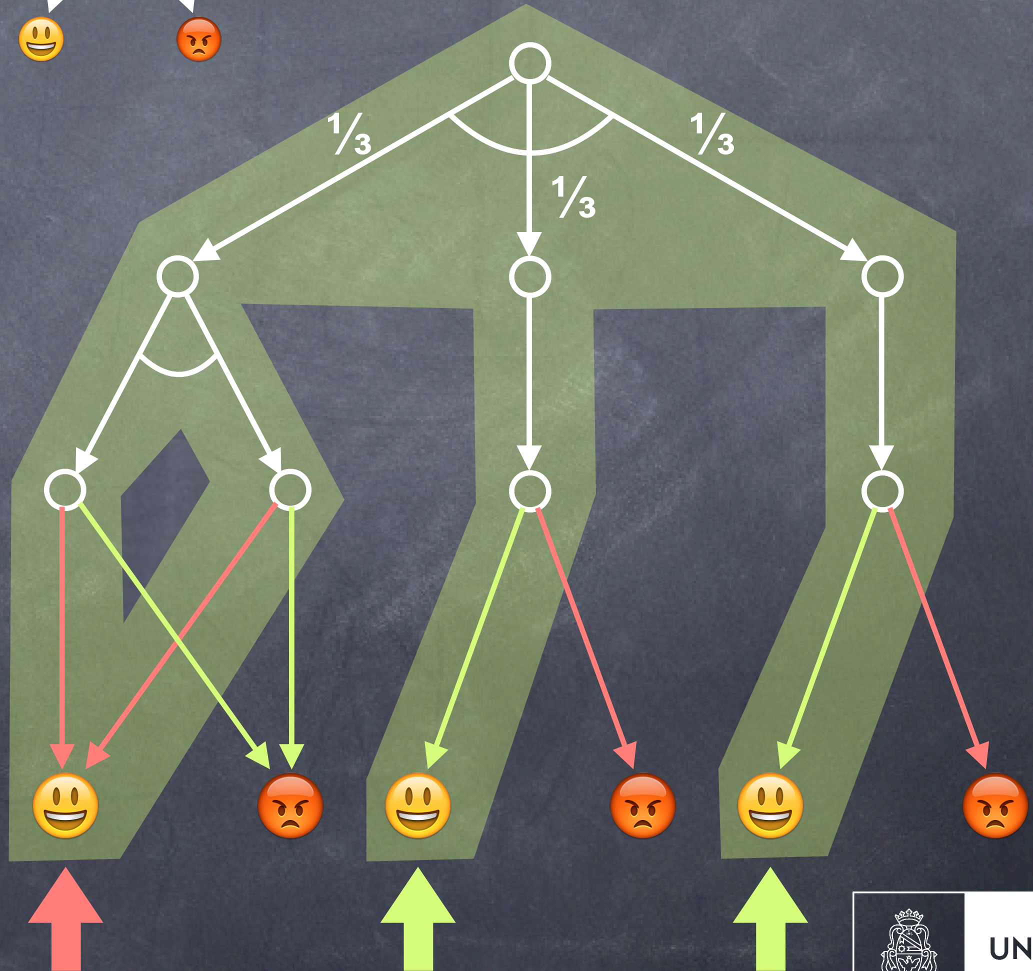
Little knowledge about other processes internal state
Local decisions can **only** be taken based on local knowledge
c!
o?
s!
k!
c?
1/3
1/3
1/3
o!
o!
o!
s?
k?
k?
s?
s?
k?
s?
k?
?
You
||
Monty Hall



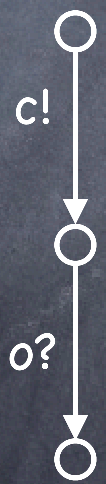
- A distributed scheduler is a scheduler that respects the local decisions of each component.
- Local decisions are only taken with the information available to each component.





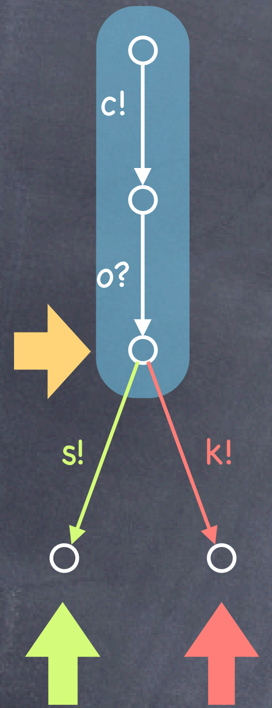


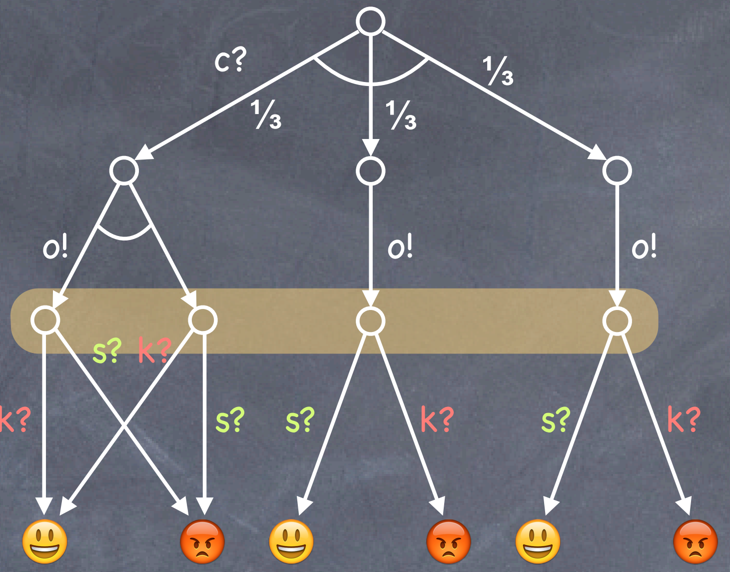
Two different choices with the same local knowledge!!



CONICET

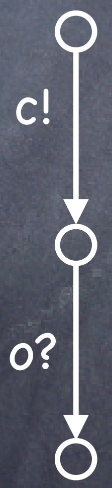
UNC

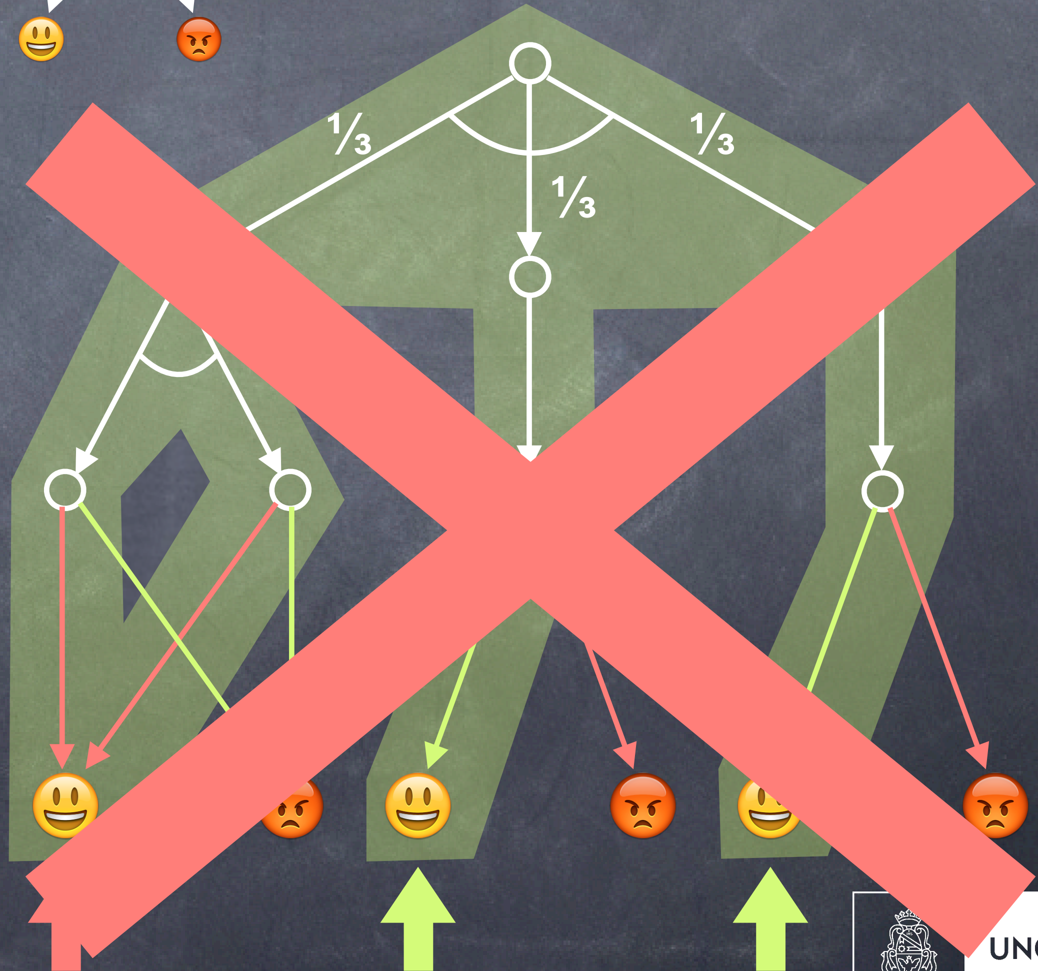




Any acceptable scheduler can do either "keep" or "switch" but not both

Two different choices with the same local knowledge!!





Probabilistic I/O automata

$(S, \bar{s}, L, \longrightarrow)$

- set of states
- initial state, $\bar{s} \in S$
- set of labels partitioned in inputs (I) and outputs (O)
- $\longrightarrow \in S \times L \times \text{Dist}(S)$ is the (probabilistic) transition relation

input enabled: $\forall a \in I : s \xrightarrow{a}$

label deterministic: $\forall a \in L : (s \xrightarrow{a} \mu' \wedge s \xrightarrow{a} \mu'') \to \mu' = \mu''$

CONICET

UNC

Composition of PIOA

- Two PIOA A_1, A_2 are compatible if $O_1 \cap O_2 = \varnothing$.
- Their parallel composition is defined by

$$A_1 \,||\, A_2 = (S_1 \times S_2, (\bar{s}_1, \bar{s}_2), L_1 \cup L_2, \to)$$

- with $O = O_1 \cup O_2$ and $I = (L_1 \cup L_2) \setminus O$, and

$$\frac{s_1 \xrightarrow{a} \mu_1}{(s_1, s_2) \xrightarrow{a} \mu_1 \times \delta_{s_2}} \quad a \in L_1 \setminus L_2$$

$$\frac{s_1 \xrightarrow{a} \mu_1 \qquad s_2 \xrightarrow{a} \mu_2}{(s_1, s_2) \xrightarrow{a} \mu_1 \times \mu_2} \quad a \in L_1 \cap L_2$$

Because of compatibility, at most one component produces an output in the composed transition

Extends to multiple components as expected

CONICET

Execution of PIOA

An execution fragment of a PIOA is a sequence

$$s_0 a_0 \mu_0 s_1 a_1 \mu_1 s_2 \ldots s_{m-1} a_{m-1} \mu_{m-1} s_m$$

such that $s_i \xrightarrow{a_i} \mu_i$ and $\mu_i(s_{i+1}) > 0$

CONICET

UNC

Schedulers

- A scheduler is a mapping from execution fragments to distributions on transitions enabled in the current state.
- Two steps to construct distributed schedulers:
  1. choose the active component A_i (i.e. the one that will produce an output),
  2. let A_i choose one output transition according to the local knowledge (suppose its label is a).
- All other A_j matching a (as an input) will do so in a parallel composition (ensured by input enabledness and determinism)

CONICET

UNC

Sch

Schedules output transitions provided this component is chosen to execute.

- For each component A_i we consider an output scheduler $\Theta_i : \mathsf{Frag}_i \to \mathsf{Dist}(O_i)$, s.t.

$$\Theta_i(\sigma)(a) > 0 \quad \text{implies} \quad \mathsf{last}(\sigma) \xrightarrow{a}_i$$

- For the system $A_1 \,||\cdots||\, A_n$ we define the interleaving scheduler $\mathcal{I} : \mathsf{Frag} \to \mathsf{Dist}(\{1,\ldots,n\})$, s.t.

$$\mathcal{I}(\sigma)(i) > 0 \quad \text{implies} \quad \exists a \in O_i : \mathsf{last}(\sigma) \xrightarrow{a}$$

Selects randomly the component that will execute an output

CONICET

Projection of an execution

The projection on a compnent A_i of an execution fragment σ of a system $A_1 \,||\cdots||\, A_n$ is defined inductively by

$$[(\bar{s}_1, \ldots, \bar{s}_n)]_i \;=\; \bar{s}_i$$

$$[\sigma\, a\, (\mu_1 \times \cdots \times \mu_n)\, (s_1, \ldots, s_n)]_i \;= \;\begin{cases} [\sigma]_i\, a\, \mu_i\, s_i & \text{if } a \in L_i \\ [\sigma]_i & \text{if } a \notin L_i \end{cases}$$

It defines the idea of "local knowledge"

UNC

Distributed Scheduler

A distributed schedulers is a mapping

$$\eta : \mathsf{Frag} \to \mathsf{Dist}(O)$$

s.t. there is a family of output schedulers $\{\Theta_i\}_i$ and an interleaving scheduler $\mathcal{I}$ so that for all $\sigma \in \mathsf{Frag}$:

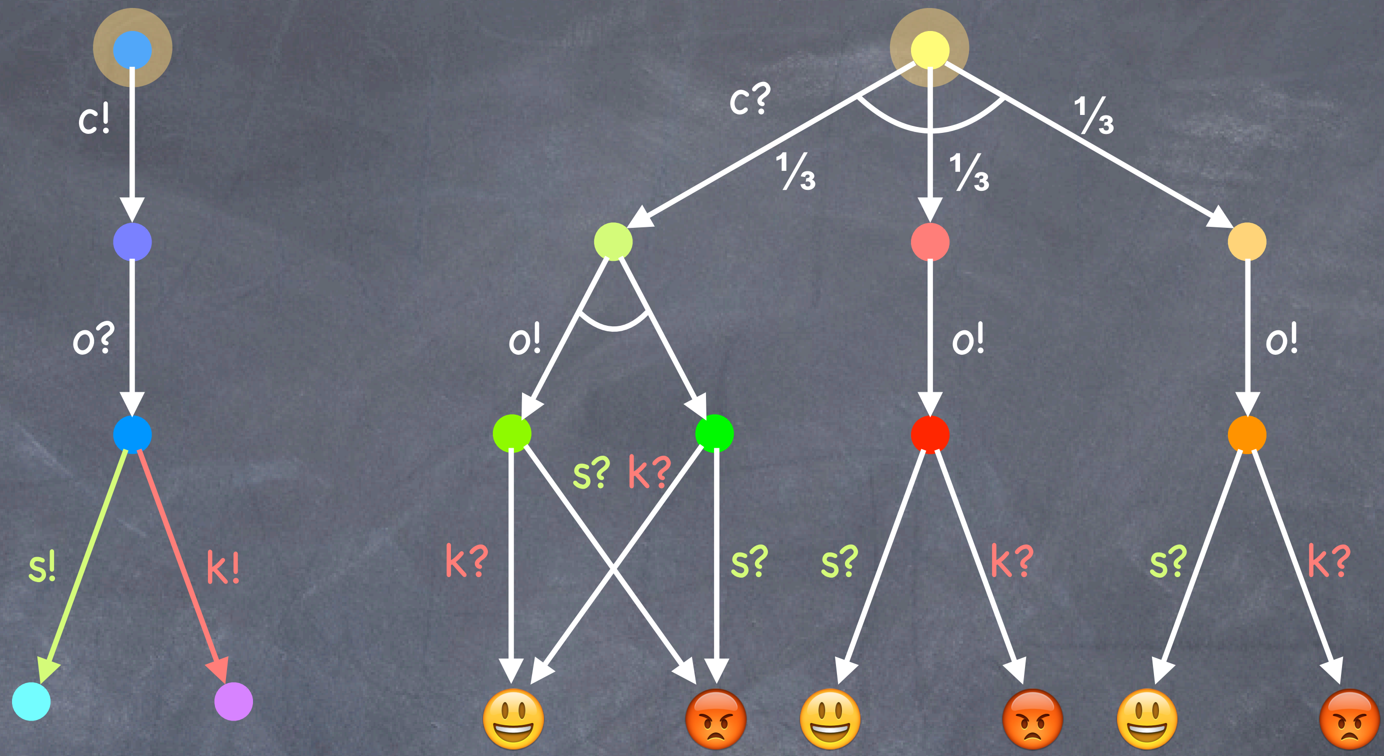
$$\eta(\sigma)(a) = \textstyle\sum_{i=1}^{n} \mathcal{I}(\sigma)(i) \cdot \Theta_i(\underline{[\sigma]_i})(a)$$

$$= \mathcal{I}(\sigma)(j) \cdot \Theta_j(\underline{[\sigma]_j})(a) \qquad \text{provided } a \in O_j$$

CONICET

UNC

Example revisited

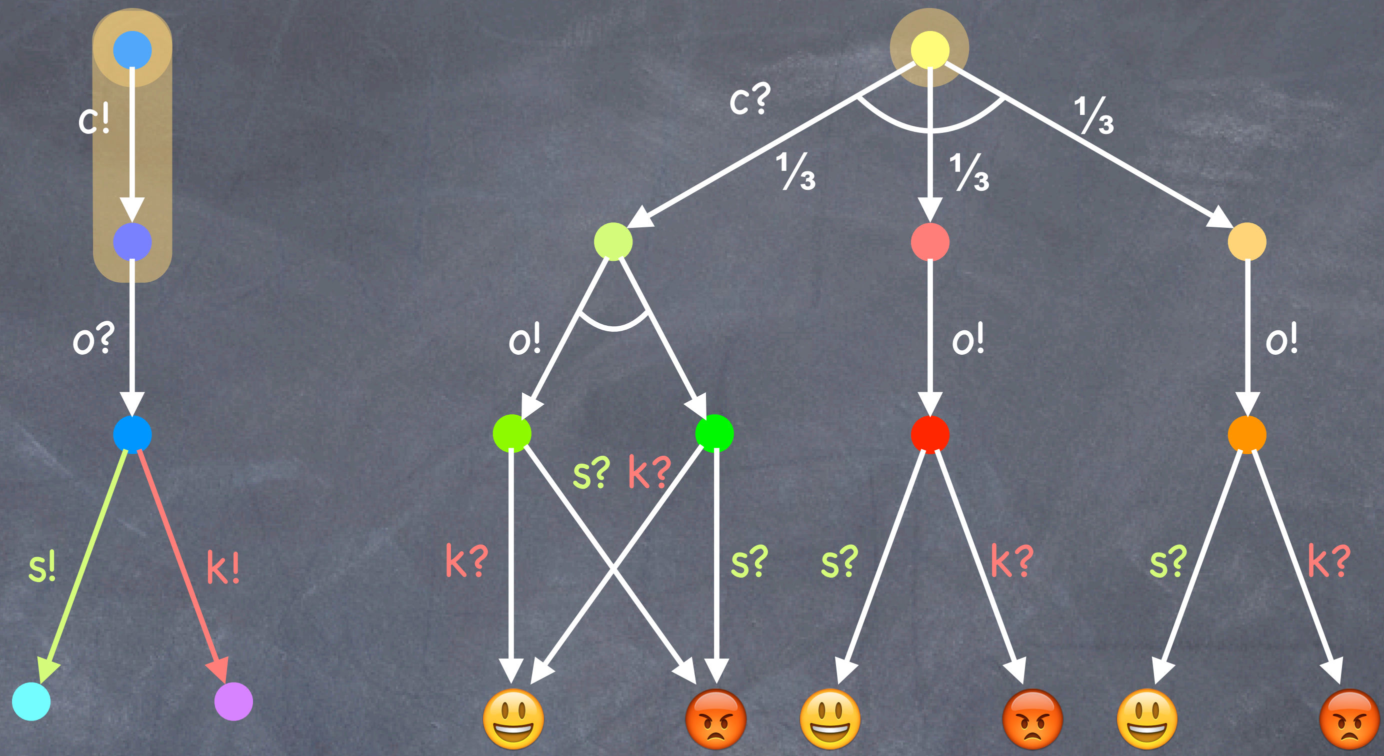


$\mathcal{I}$ ((•,•)) = You

CONICET

UNC

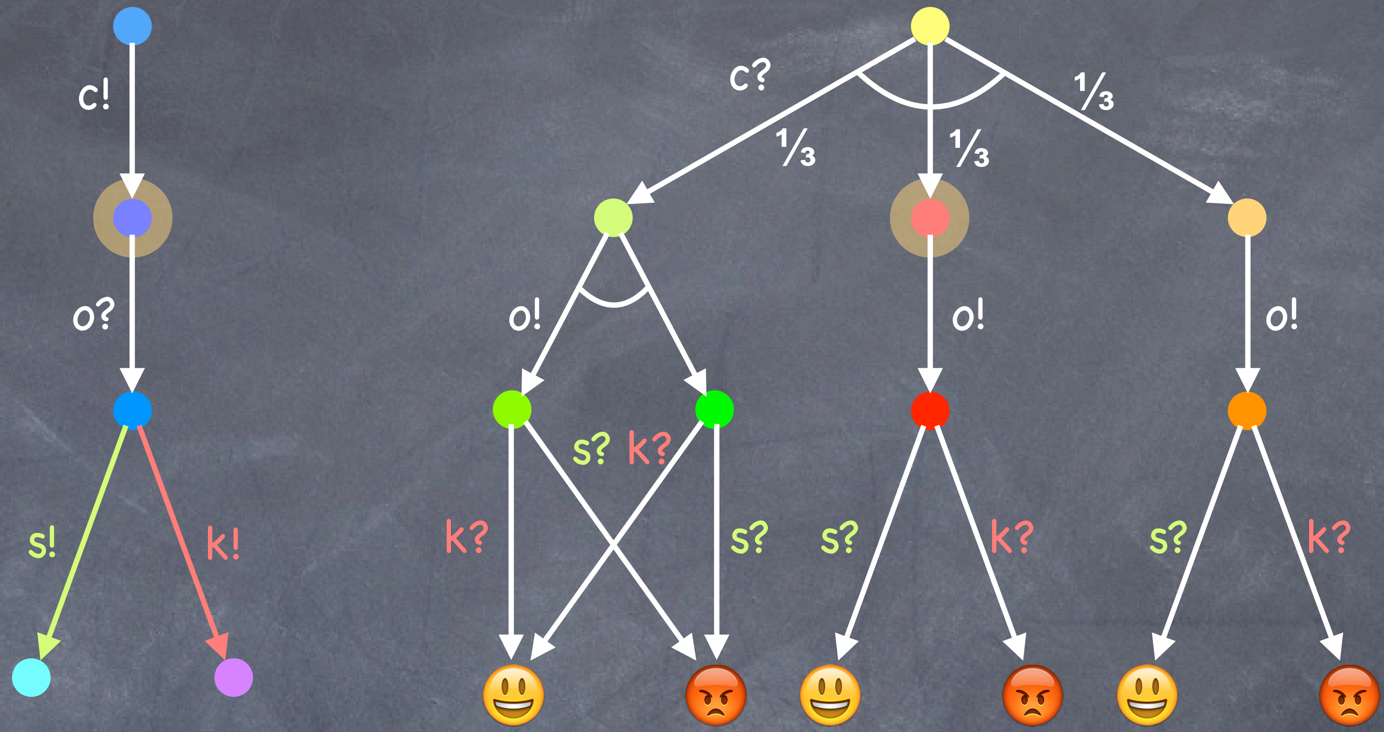
Example revisited



$\mathcal{I}$ ((•,•)) = You

Θ_Y ([(•,•)]$_Y$) = Θ_Y (•) = c!

Example revisited

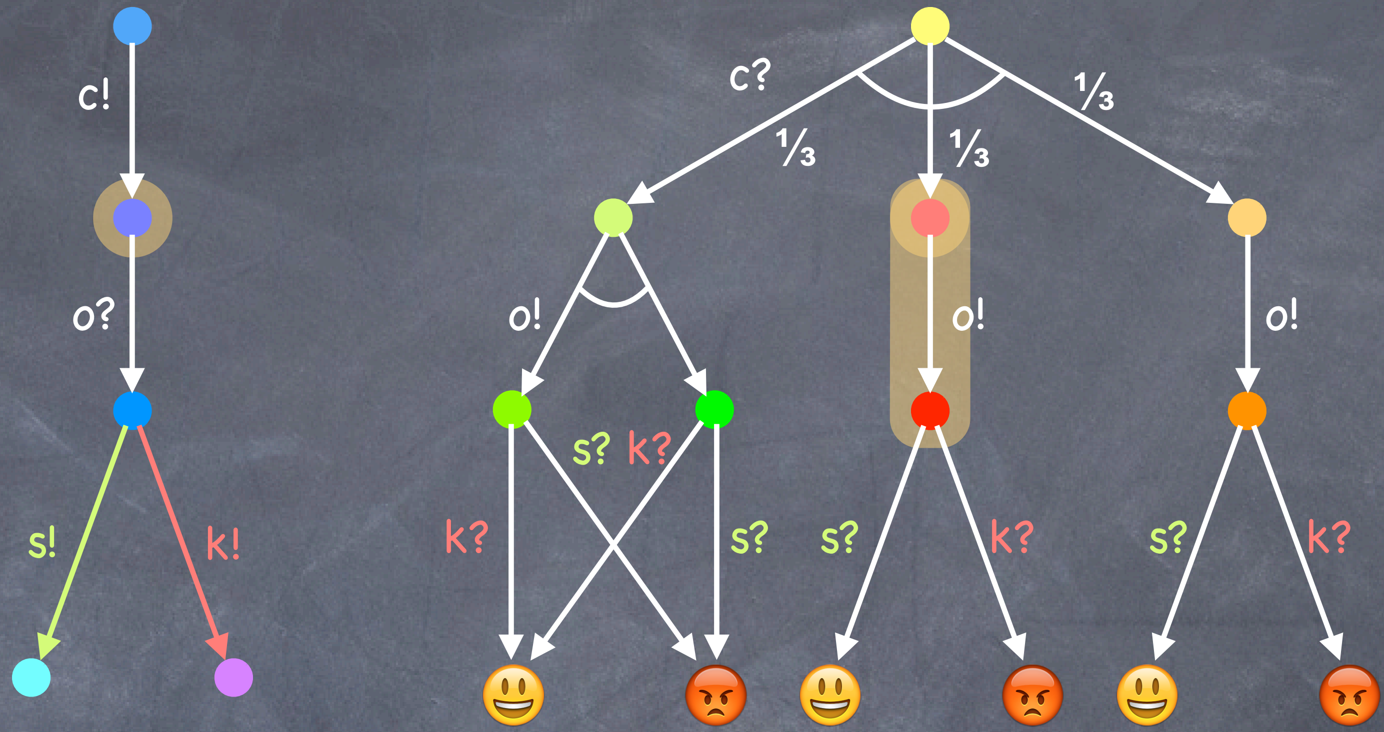


$\mathcal{I}$ ((•,•)) = You

Θ_Y ([(•,•)]$_Y$) = Θ_Y (•) = c!

$\mathcal{I}$ ((•,•)c(•,•)) = MH

Example revisited



$\mathcal{I}$ ((•,•)) = You

Θ_Y ([(•,•)]$_Y$) = Θ_Y (•) = c!

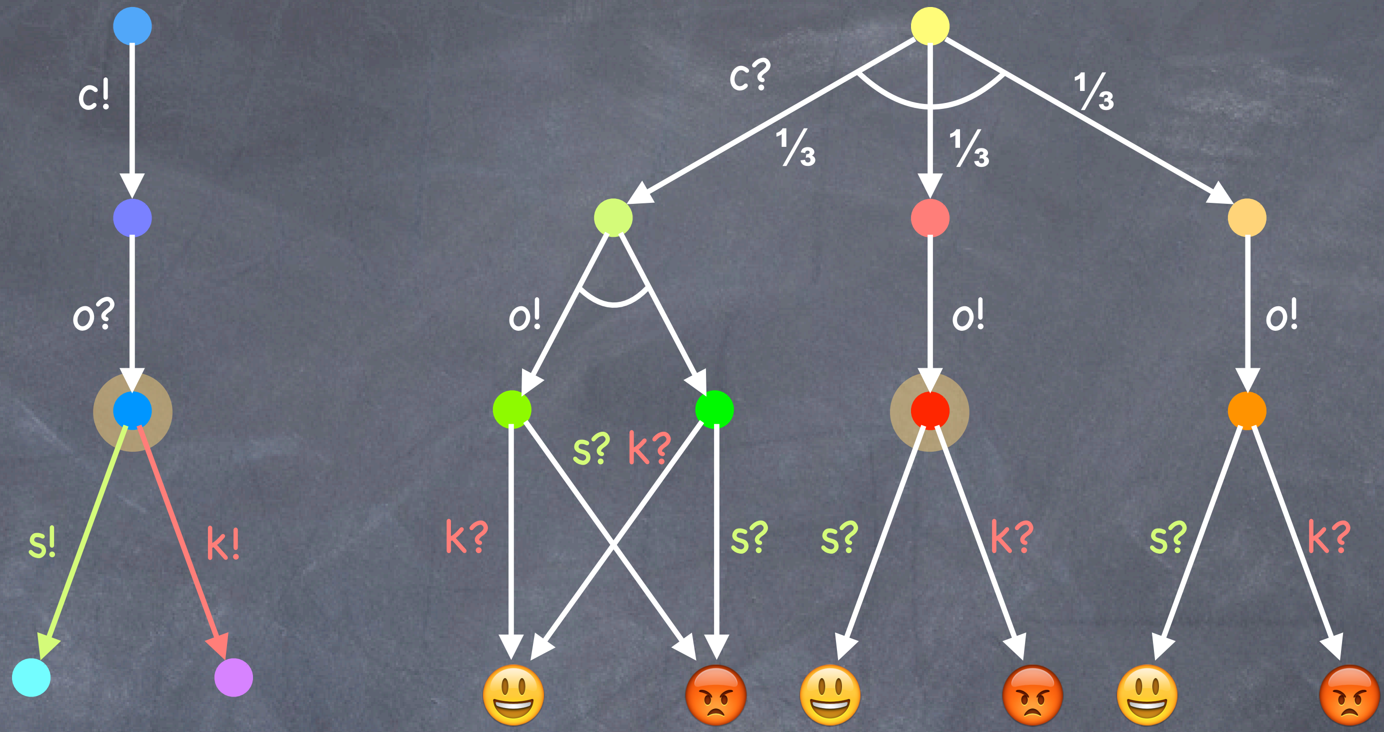
$\mathcal{I}$ ((•,•)c(•,•)) = MH

Θ_{MH} ([(•,•)c(•,•)]$_{MH}$) = Θ_{MH} (•c•) = o!

CONICET

UNC

Example revisited



$\mathcal{I}(\,(\bullet,\bullet)\,) = \text{You}$

$\Theta_Y(\,[(\bullet,\bullet)]_Y\,) = \Theta_Y(\bullet) = c!$

$\mathcal{I}(\,(\bullet,\bullet)c(\bullet,\bullet)) = \text{MH}$

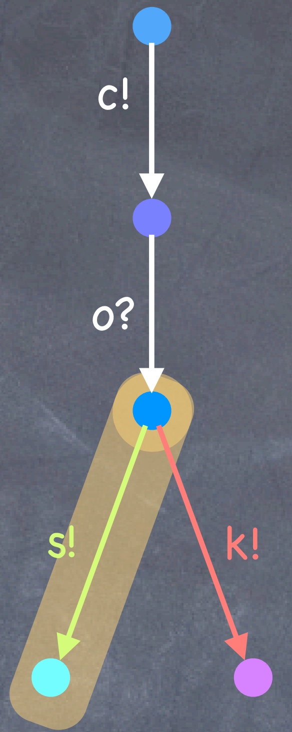
$\Theta_{MH}(\,[(\bullet,\bullet)c(\bullet,\bullet)]_{MH}\,) = \Theta_{MH}(\bullet c \bullet) = o!$

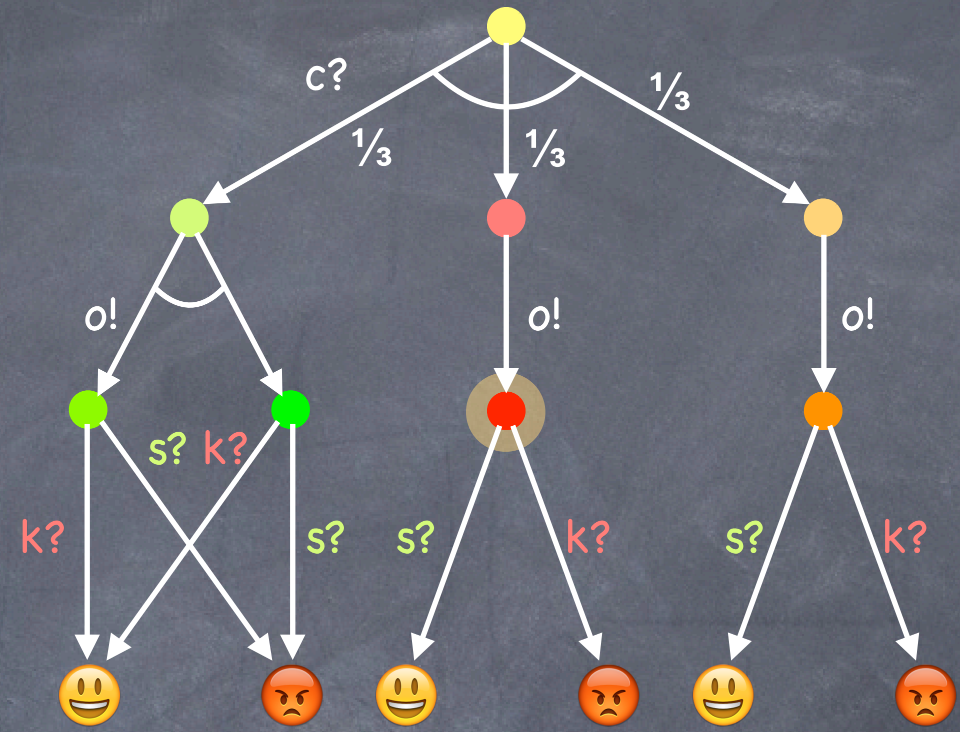
$\mathcal{I}(\,(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)\,) = \text{You}$

CONICET

UNC

Example revisited





$\mathcal{I}\,(\,(\bullet,\bullet)\,) = \text{You}$

$\Theta_{Y}\,(\,[(\bullet,\bullet)]_{Y}\,) = \Theta_{Y}\,(\bullet) = c!$

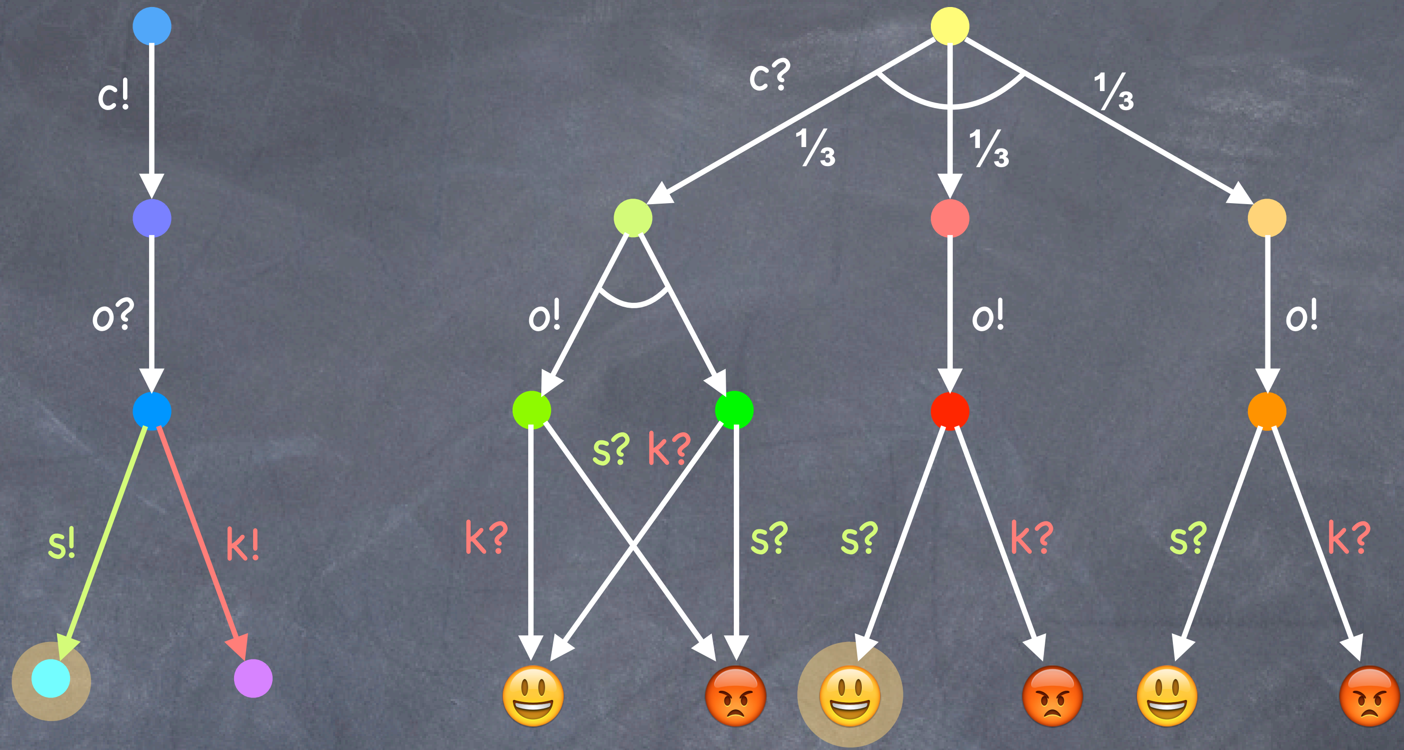
$\mathcal{I}\,(\,(\bullet,\bullet)c(\bullet,\bullet)) = \text{MH}$

$\Theta_{MH}\,(\,[(\bullet,\bullet)c(\bullet,\bullet)]_{MH}\,) = \Theta_{MH}\,(\bullet c \bullet) = o!$

$\mathcal{I}\,(\,(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)\,) = \text{You}$

$\Theta_{Y}\,(\,[(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)]_{Y}\,) = \Theta_{Y}\,(\bullet c \bullet o \bullet) = s!$

Example revisited



$\mathcal{I}\,(\,(\bullet,\bullet)\,) = \mathrm{You}$

$\Theta_Y\,(\,[(\bullet,\bullet)]_Y\,) = \Theta_Y\,(\bullet) = c!$

$\mathcal{I}\,(\,(\bullet,\bullet)c(\bullet,\bullet)) = \mathrm{MH}$

$\Theta_{MH}\,(\,[(\bullet,\bullet)c(\bullet,\bullet)]_{MH}\,) = \Theta_{MH}\,(\bullet c \bullet) = o!$

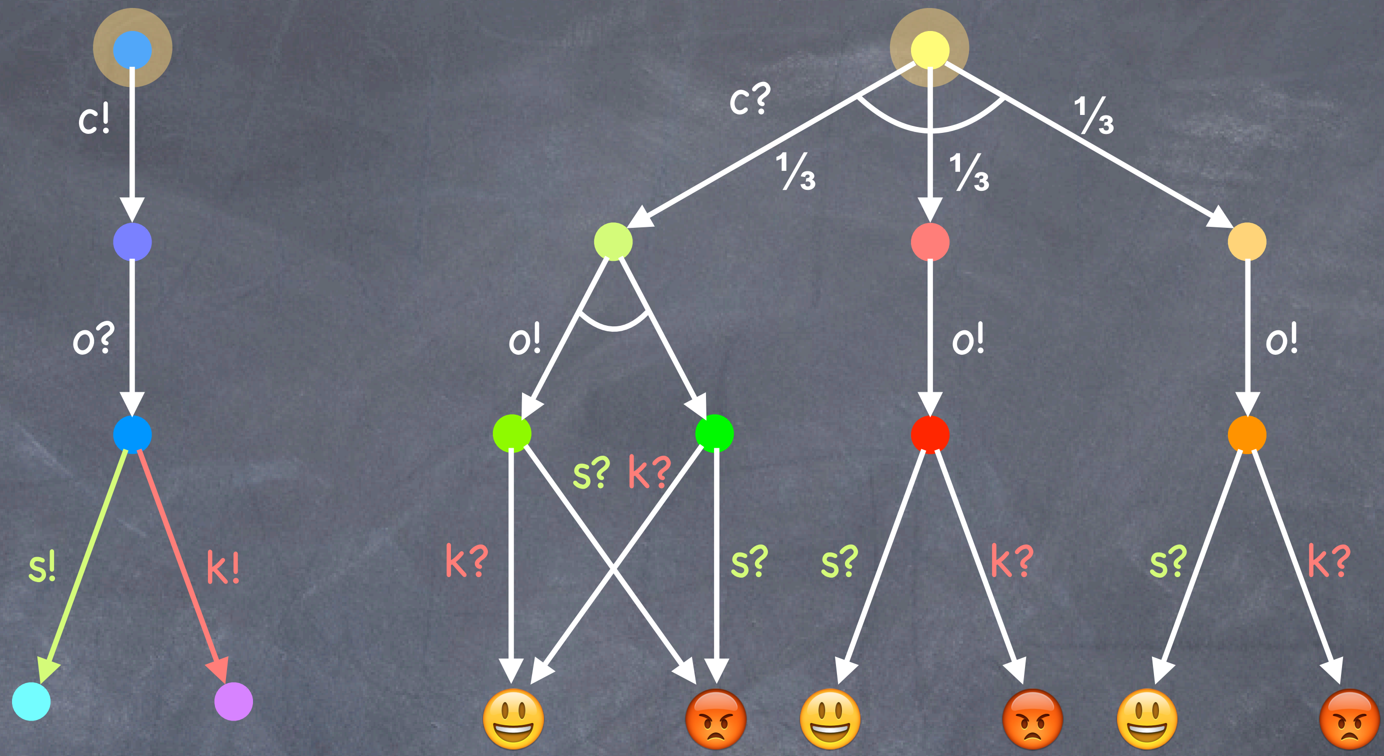
$\mathcal{I}\,(\,(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)\,) = \mathrm{You}$

$\Theta_Y\,(\,[(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)]_Y\,) = \Theta_Y\,(\bullet c \bullet o \bullet) = s!$

CONICET

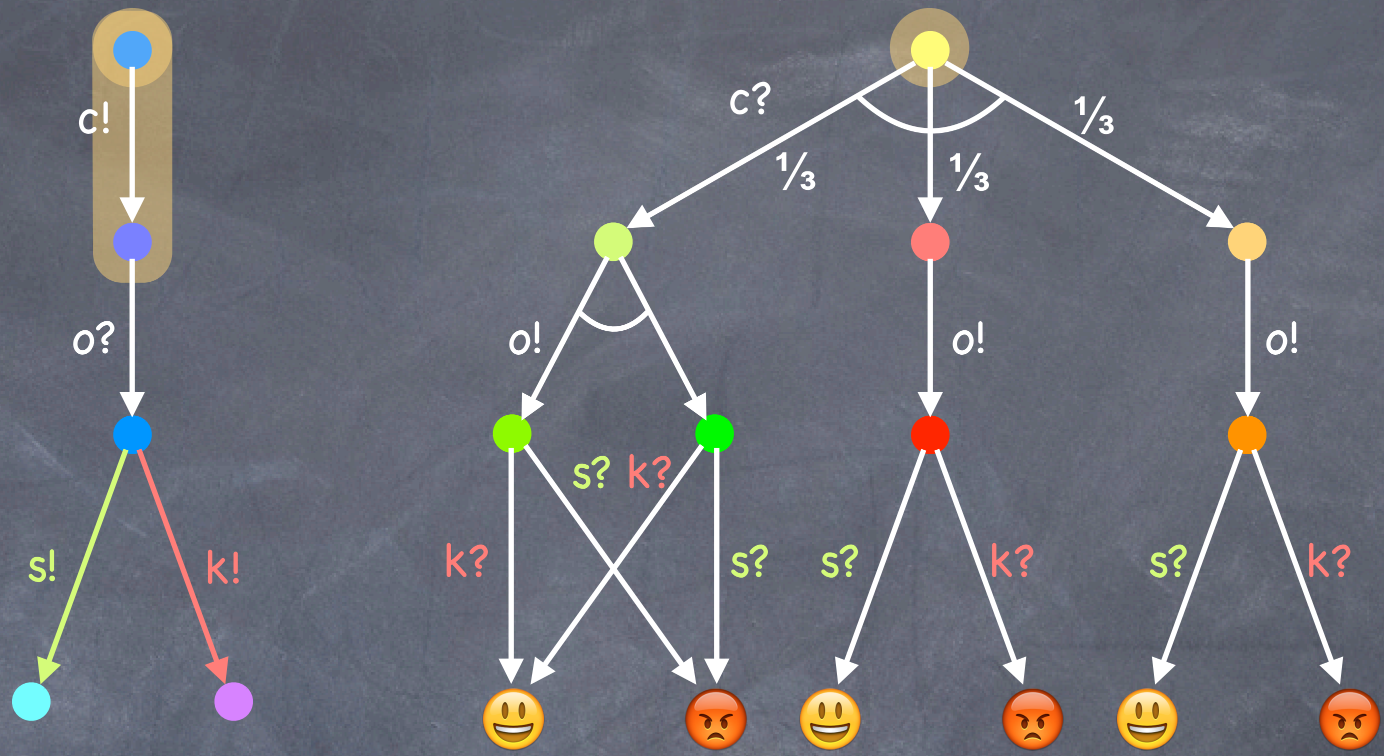
UNC

Example revisited



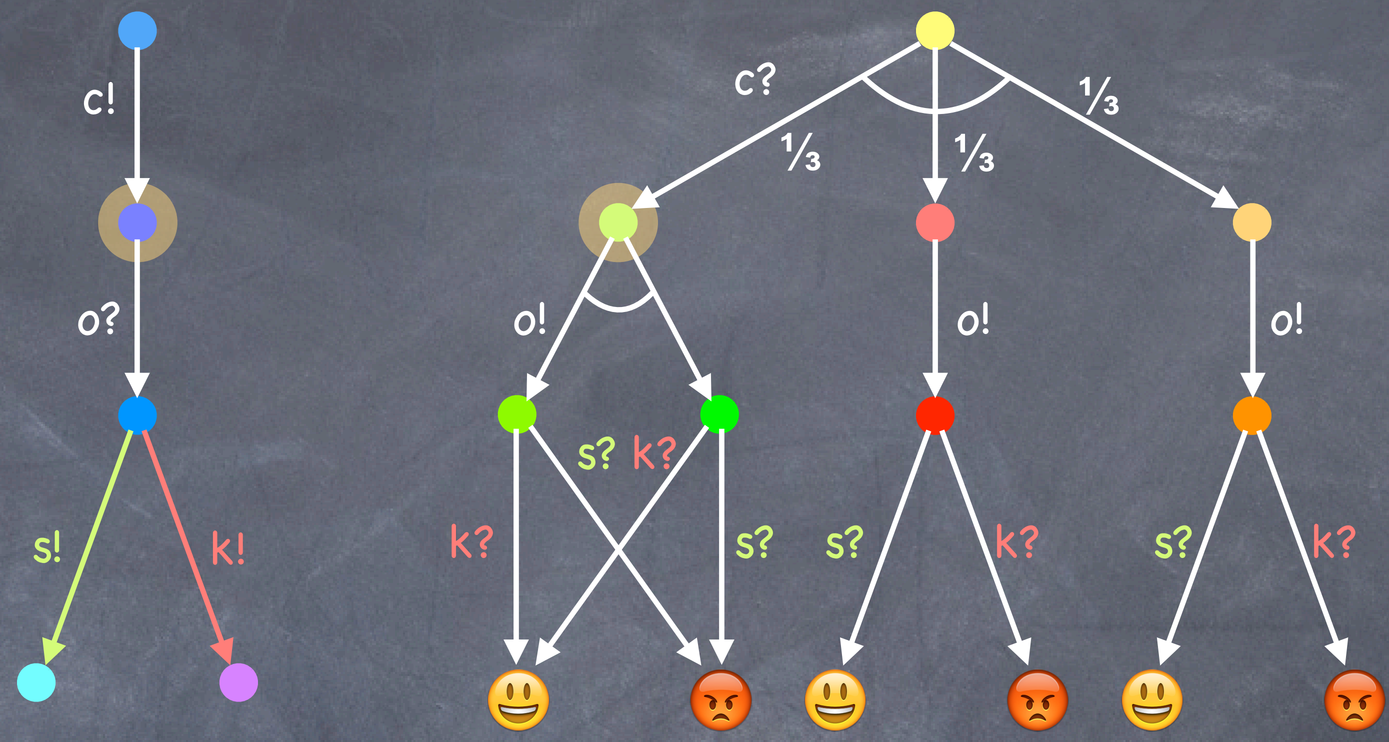
Θ_Y ([(•,•)c(•,•)o(•,•)]$_Y$) = Θ_Y (•c•o•) = s!

Example revisited



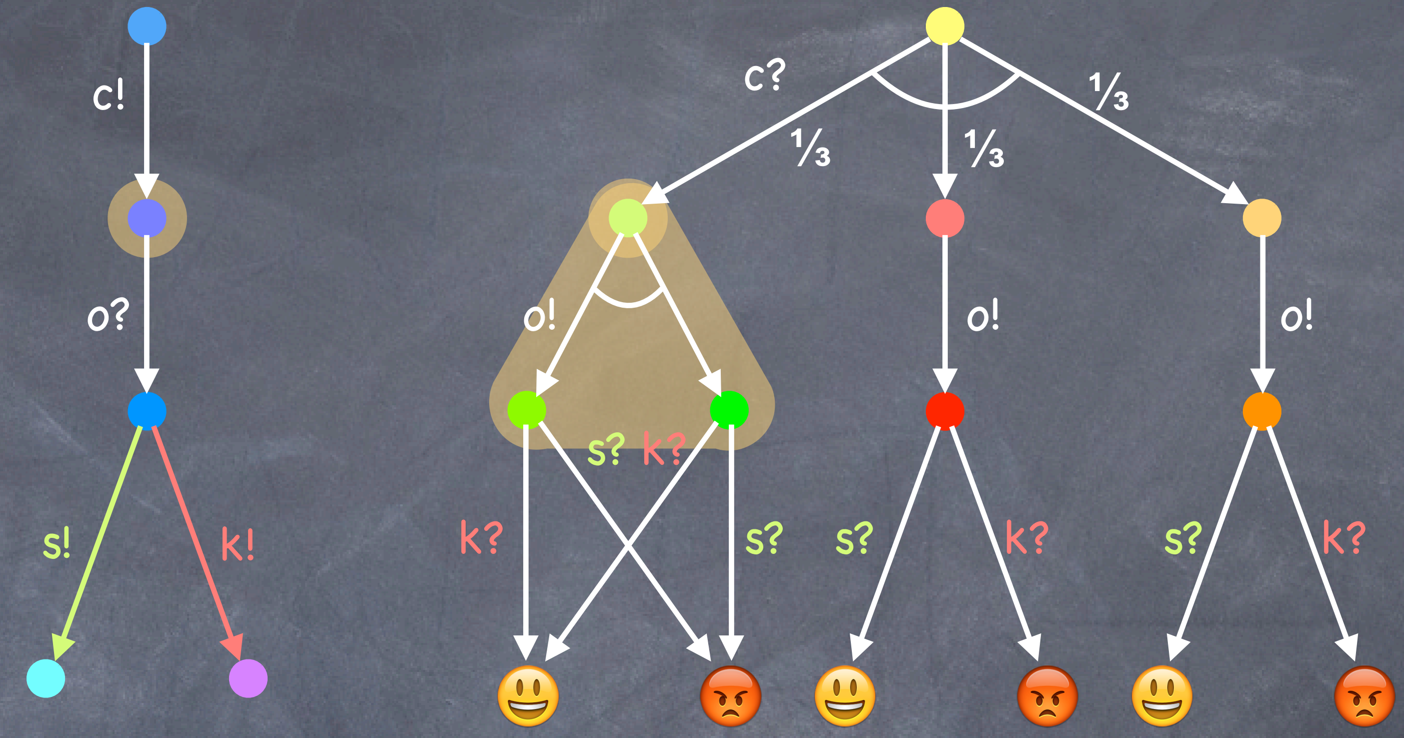
Θ_Y ($[(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)]_Y$) = Θ_Y ($\bullet c \bullet o \bullet$) = s!

Example revisited



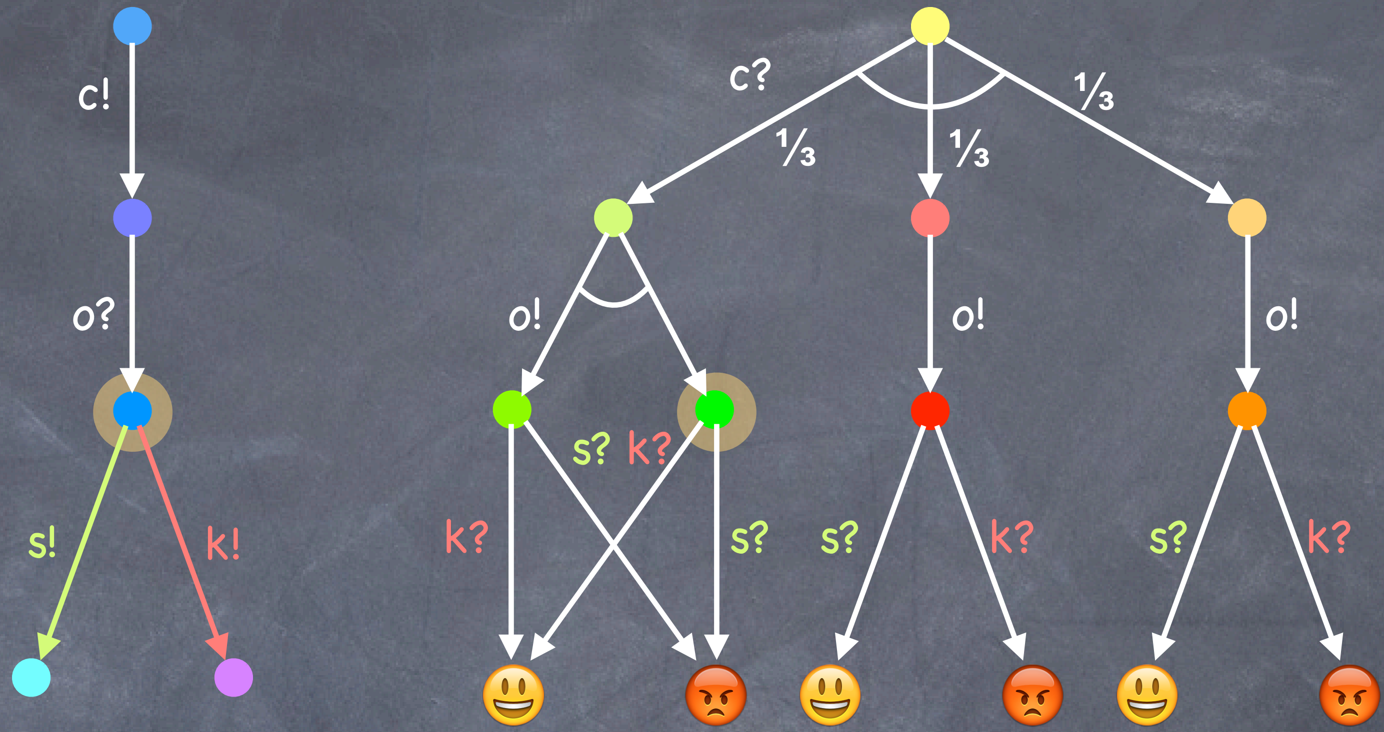
$$\Theta_Y \left([(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)]_Y \right) = \Theta_Y \left(\bullet c \bullet o \bullet\right) = s!$$

Example revisited



$$\Theta_Y\left(\,[(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)]_Y\,\right) = \Theta_Y\left(\bullet c \bullet o \bullet\right) = s!$$

Example revisited



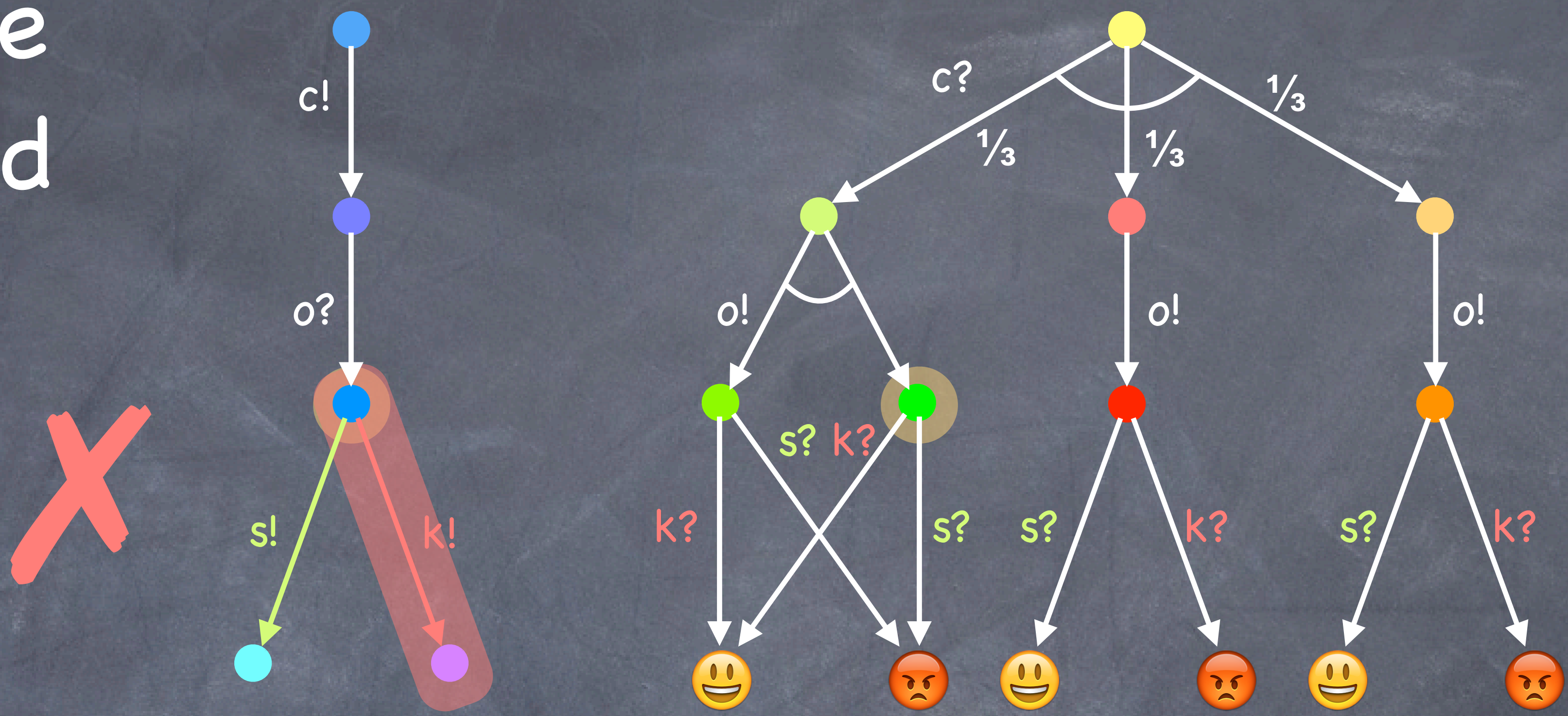
Θ_Y ([(•,•)c(•,•)o(•,•)]$_Y$) = $\boxed{\Theta_Y \text{ (•c•o•)}}$ = s!

Θ_Y ([(•,•)c(•,•)o(•,•)]$_Y$) = $\boxed{\Theta_Y \text{ (•c•o•)}}$ =

CONICET

UNC

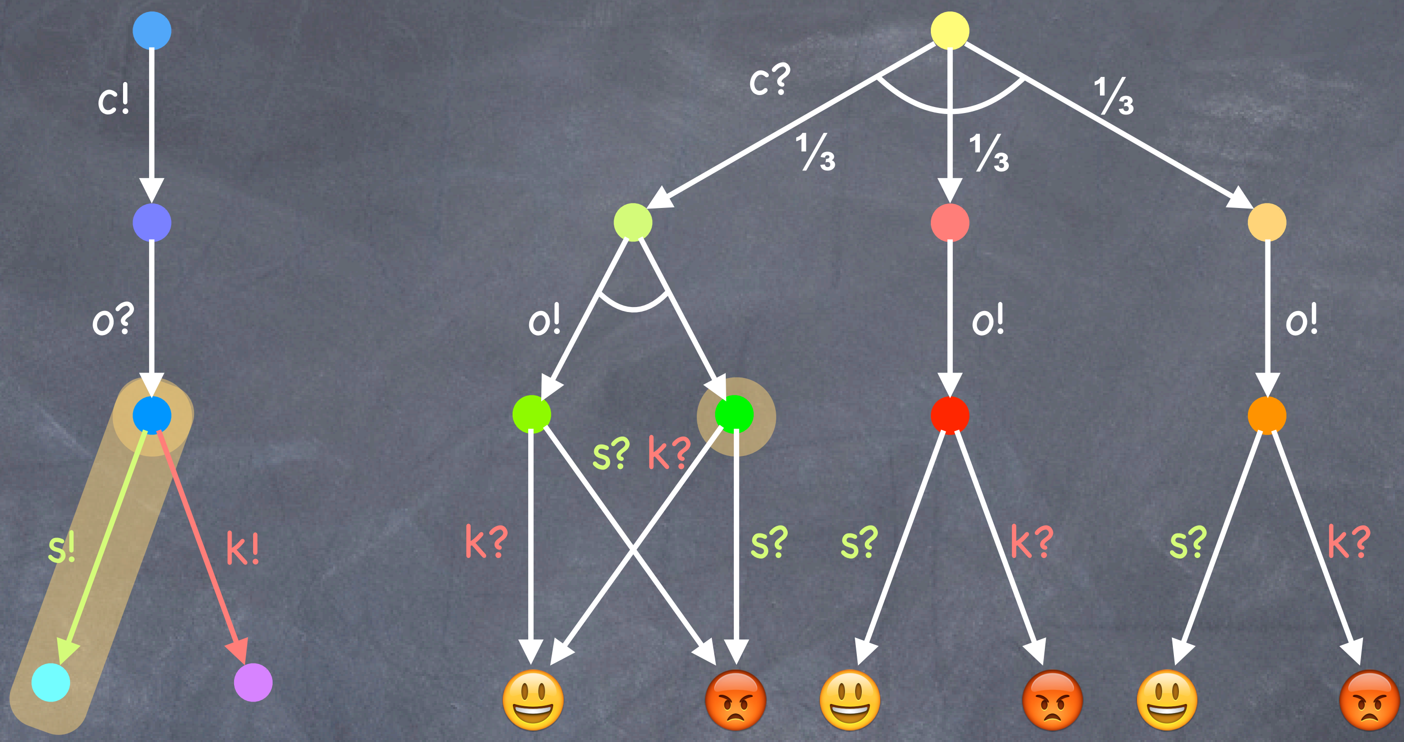
Example revisited



Θ_Y ([(•,•)c(•,•)o(•,•)]$_Y$) = Θ_Y (•c•o•) = s!

Θ_Y ([(•,•)c(•,•)o(•,•)]$_Y$) = Θ_Y (•c•o•) =

Example revisited



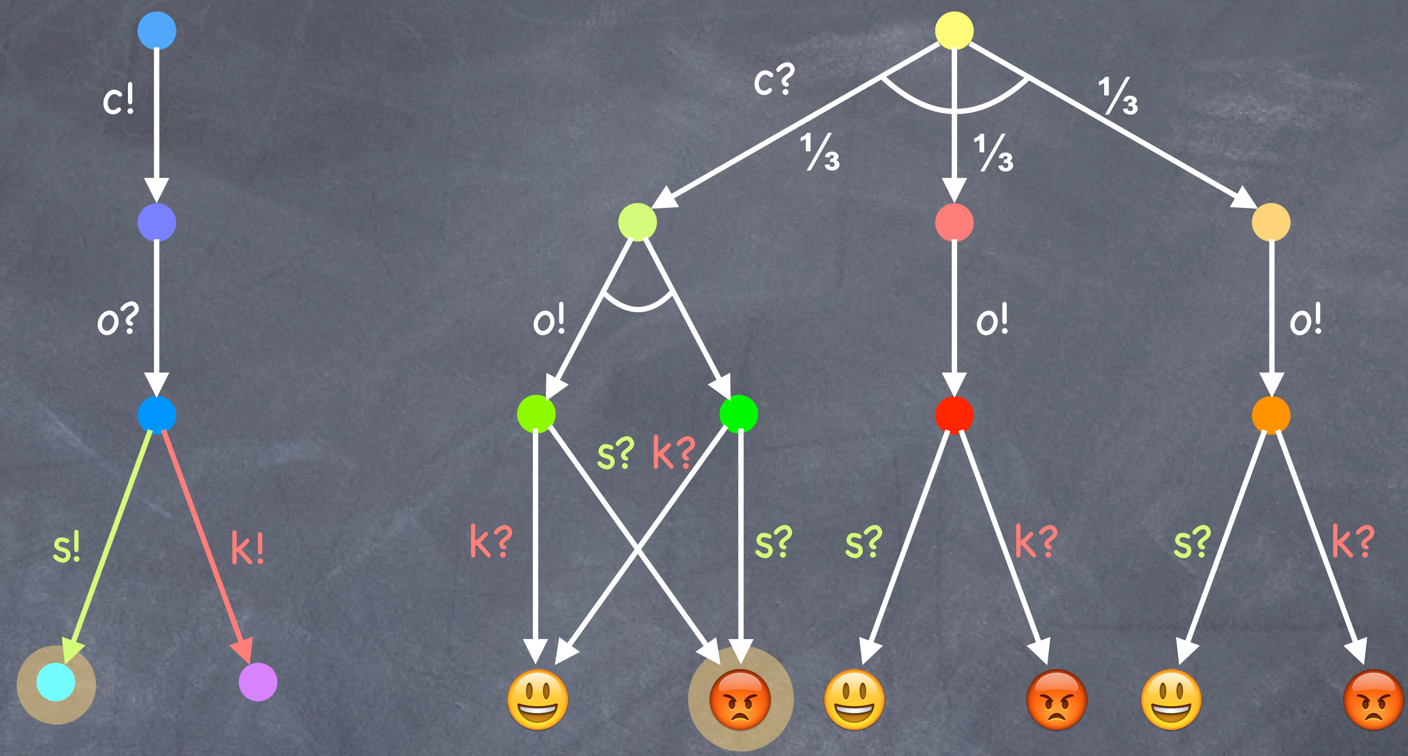
$$\Theta_Y\left(\,[(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)]_Y\,\right) = \boxed{\Theta_Y\left(\bullet c \bullet o \bullet\right)} = s!$$

$$\Theta_Y\left(\,[(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)]_Y\,\right) = \boxed{\Theta_Y\left(\bullet c \bullet o \bullet\right)} = s!$$

CONICET

UNC

Example revisited



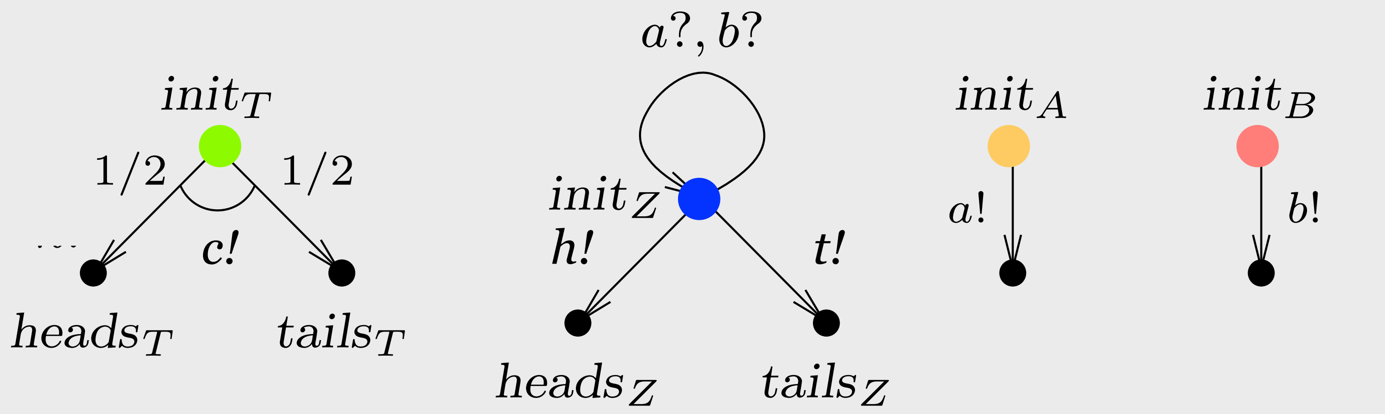
$$\Theta_Y\ (\ [(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)]_Y\ ) = \Theta_Y\ (\bullet c \bullet o \bullet) = s!$$

$$\Theta_Y\ (\ [(\bullet,\bullet)c(\bullet,\bullet)o(\bullet,\bullet)]_Y\ ) = \Theta_Y\ (\bullet c \bullet o \bullet) = s!$$

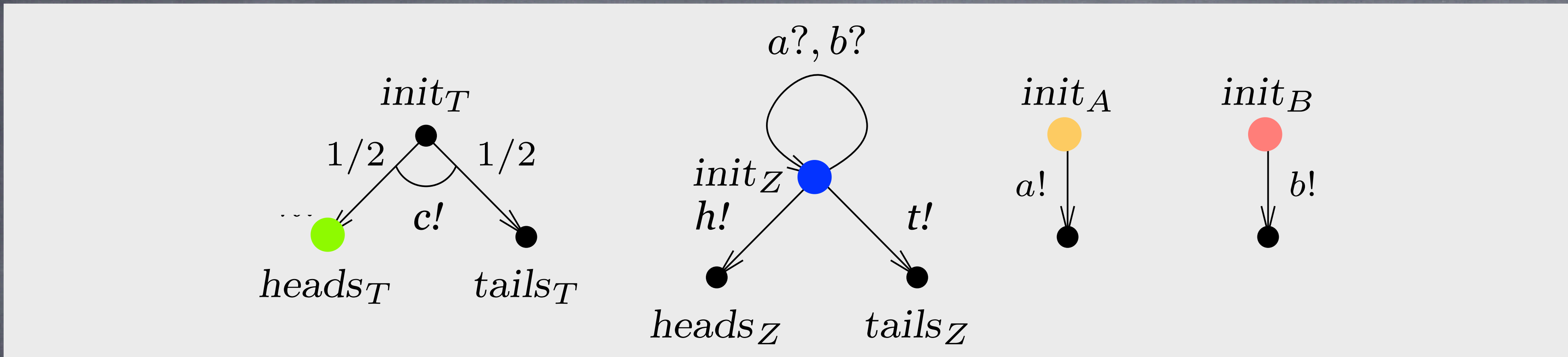
CONICET

UNC

Are distributed schedulers what we need?

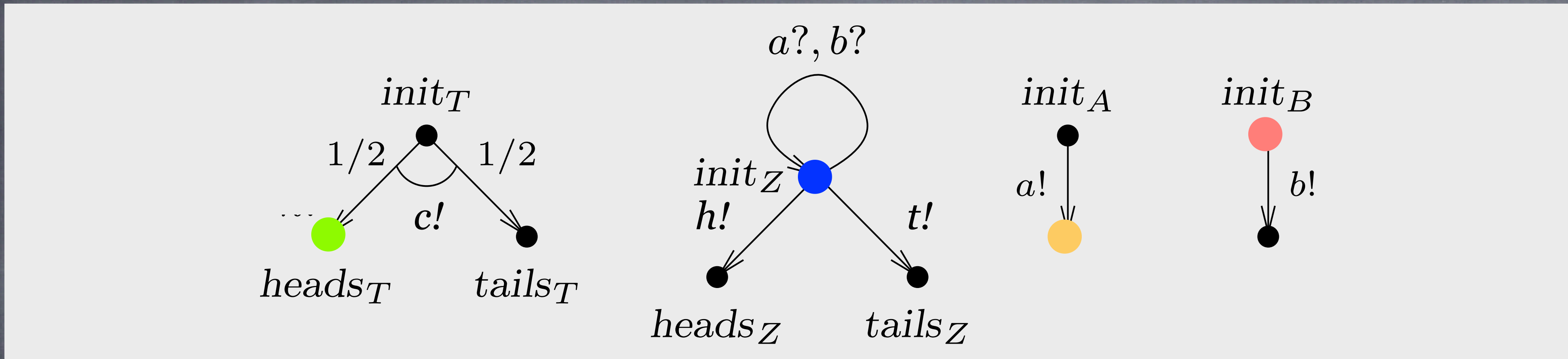


Are distributed schedulers what we need?



$\mathcal{I}\ (\ (i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\ ) = 3$

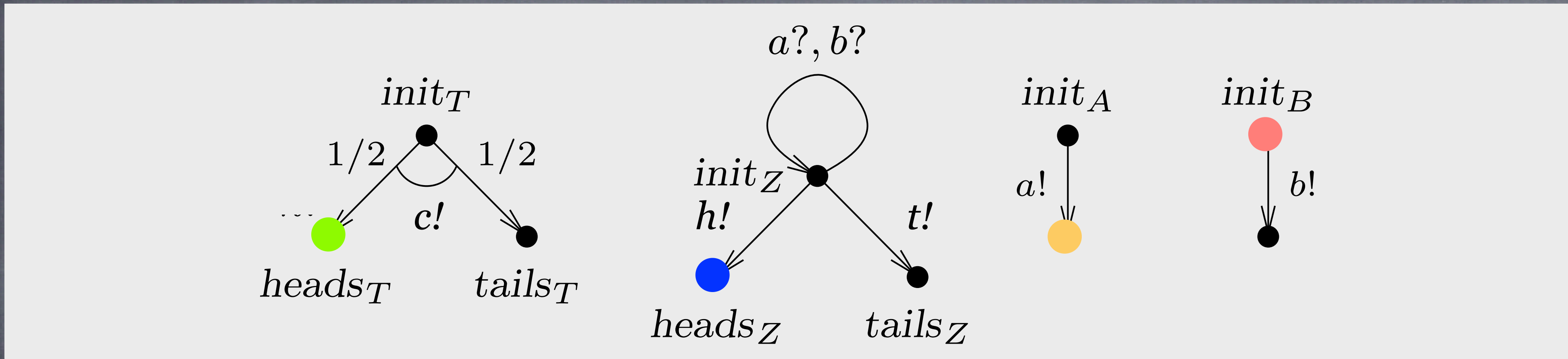
Are distributed schedulers what we need?



$\mathcal{I}$ ((i_T, i_Z, i_A, i_B) $c!$ (h_T, i_Z, i_A, i_B)) = 3

$\mathcal{I}$ ((i_T, i_Z, i_A, i_B) $c!$ (h_T, i_Z, i_A, i_B) $a!$ (h_T, i_Z, e_A, i_B)) = 2

Are distributed schedulers what we need?

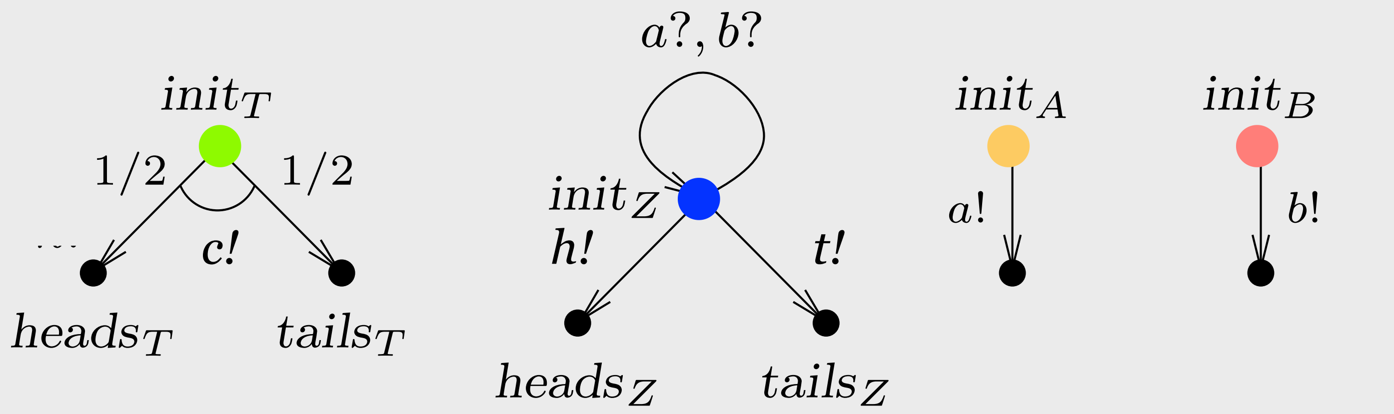


$\mathcal{I}\left((i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B) \right) = 3$

$\mathcal{I}\left((i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\ a!\ (h_T, i_Z, e_A, i_B) \right) = 2$

$\Theta_2\left([(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\ a!\ (h_T, i_Z, e_A, i_B)]_2 \right) = \Theta_2\left(i_Z\ a!\ i_Z \right) = h!$

Are distributed schedulers what we need?



$\mathcal{I}$ ((i_T, i_Z, i_A, i_B) $c!$ (h_T, i_Z, i_A, i_B)) = 3

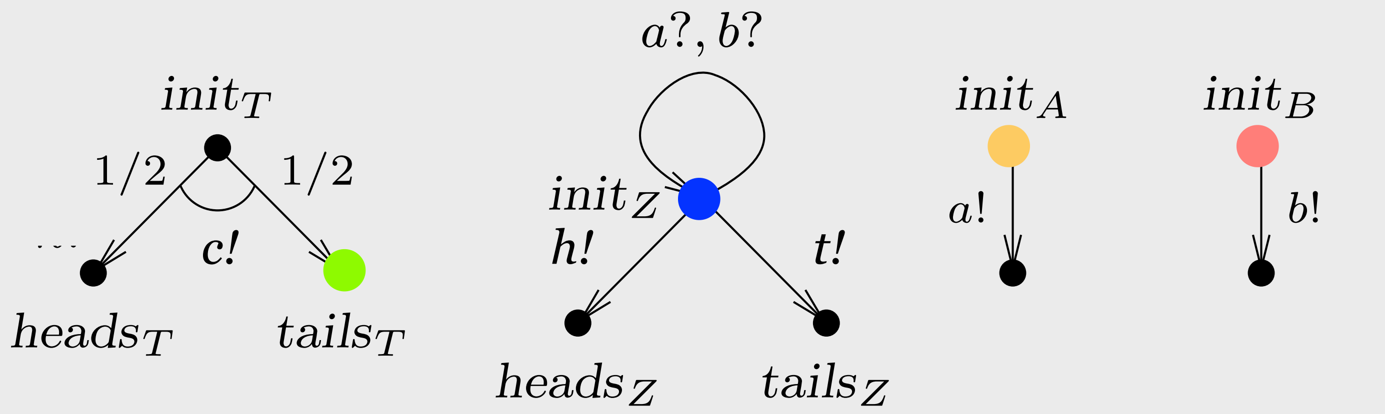
$\mathcal{I}$ ((i_T, i_Z, i_A, i_B) $c!$ (h_T, i_Z, i_A, i_B) $a!$ (h_T, i_Z, e_A, i_B)) = 2

Θ_2 ($[(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\ a!\ (h_T, i_Z, e_A, i_B)]_2$) = Θ_2 ($i_Z\ a!\ i_Z$) = $h!$

CONICET

UNC

Are distributed schedulers what we need?



$\mathcal{I}$ ((i_T, i_Z, i_A, i_B) $c!$ (h_T, i_Z, i_A, i_B)) = 3

$\mathcal{I}$ ((i_T, i_Z, i_A, i_B) $c!$ (h_T, i_Z, i_A, i_B) $a!$ (h_T, i_Z, e_A, i_B)) = 2

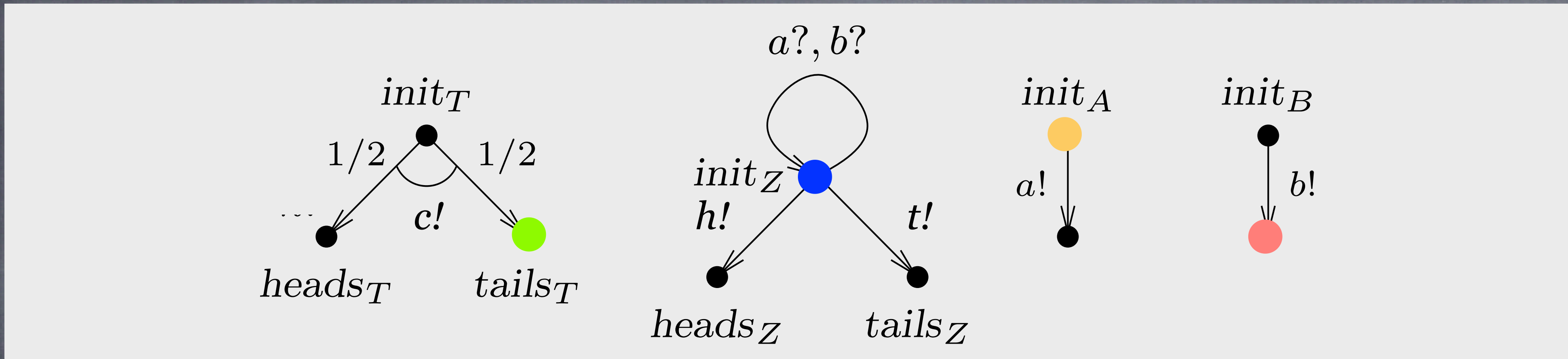
Θ_2 ($[(i_T, i_Z, i_A, i_B)$ $c!$ (h_T, i_Z, i_A, i_B) $a!$ $(h_T, i_Z, e_A, i_B)]_2$) = Θ_2 (i_Z $a!$ i_Z) = $h!$

$\mathcal{I}$ ((i_T, i_Z, i_A, i_B) $c!$ (t_T, i_Z, i_A, i_B)) = 4

CONICET

UNC

Are distributed schedulers what we need?



$\mathcal{I}\ (\ (i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\ ) = 3$

$\mathcal{I}\ (\ (i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\ a!\ (h_T, i_Z, e_A, i_B)\ ) = 2$

$\Theta_2\ (\ [(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\ a!\ (h_T, i_Z, e_A, i_B)]_2\ ) = \Theta_2\ (\ i_Z\ a!\ i_Z\ ) = h!$

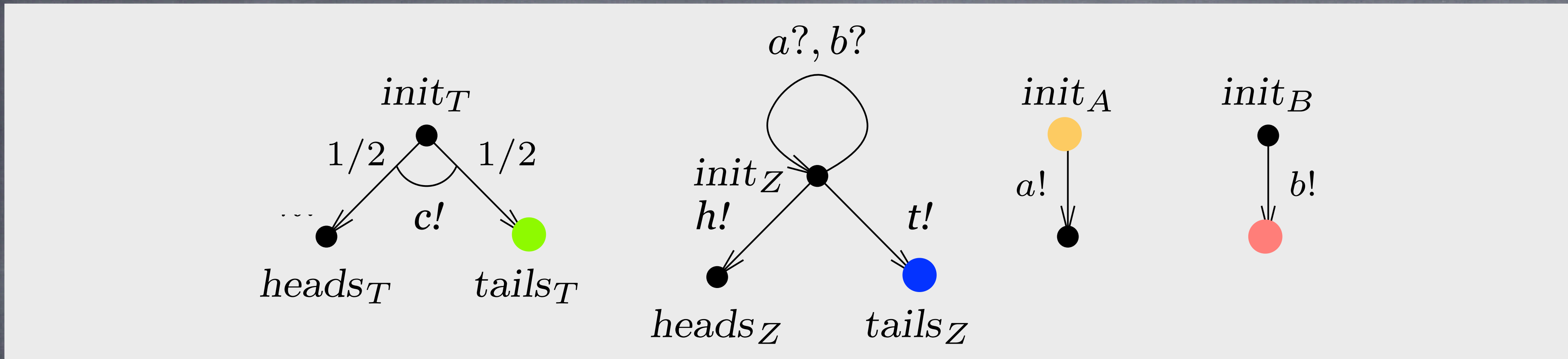
$\mathcal{I}\ (\ (i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B)\ ) = 4$

$\mathcal{I}\ (\ (i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B)\ b!\ (t_T, i_Z, i_A, e_B)\ ) = 2$

CONICET

UNC

Are distributed schedulers what we need?



$\mathcal{I}\,(\,(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\,) = 3$

$\mathcal{I}\,(\,(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\ a!\ (h_T, i_Z, e_A, i_B)\,) = 2$

$\Theta_2\,(\,[(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\ a!\ (h_T, i_Z, e_A, i_B)]_2\,) = \Theta_2\,(\,\underbrace{i_Z\ a!\ i_Z}\,) = h!$

$\neq$

$\mathcal{I}\,(\,(i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B)\,) = 4$

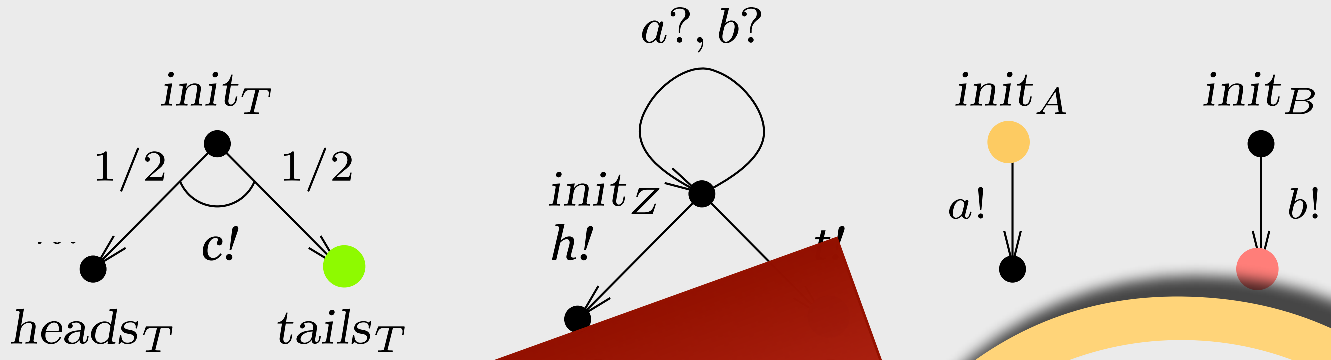
$\mathcal{I}\,(\,(i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B)\ b!\ (t_T, i_Z, i_A, e_B)\,) = 2$

$\Theta_2\,(\,[(i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B)\ b!\ (t_T, i_Z, i_A, e_B)]_2\,) = \Theta_2\,(\,\overbrace{i_Z\ b!\ i_Z}\,) = t!$

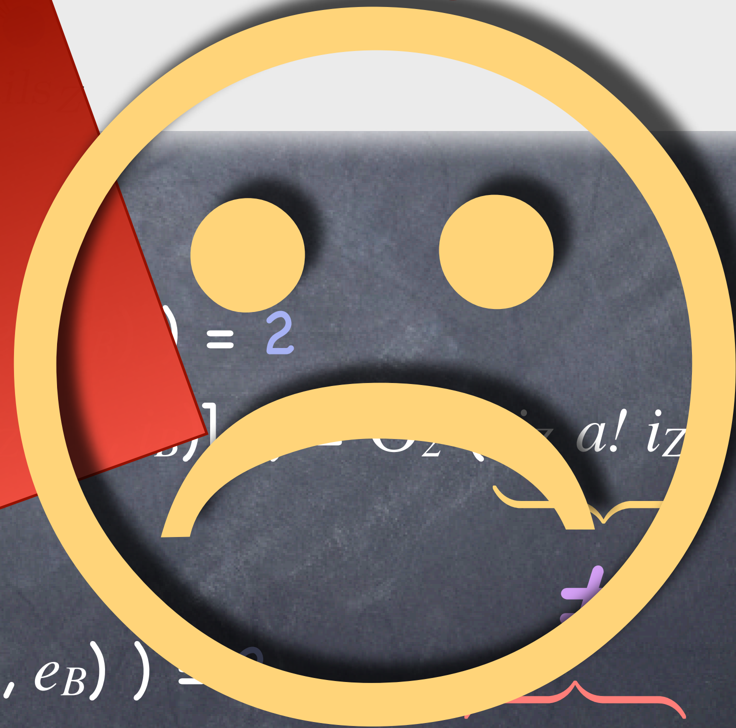
CONICET

UNC

Are distributed schedulers what we need?



MAXIMUM PROBABILIY IS 1



$\mathcal{I}$ ((i_T, i_Z, i_A ...

$\mathcal{I}$ ((...) = 2

Θ_2 ([(...]) = Θ_2 (... a! i_Z) = h!

$\mathcal{I}$ ((i_T, i_Z ... 4

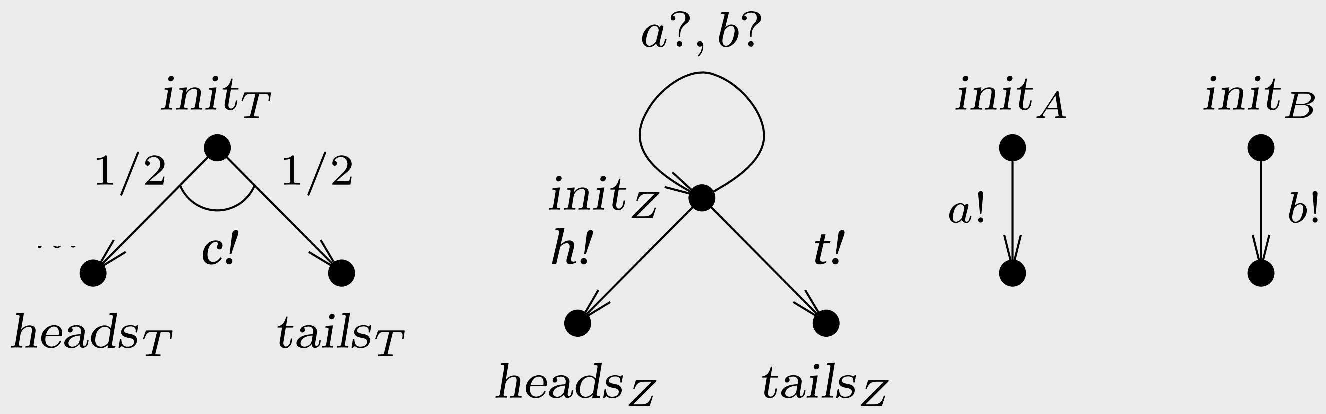
$\mathcal{I}$ ((i_T, i_Z, ... i_A, i_B) b! (t_T, i_Z, i_A, e_B)) = ...

Θ_2 ([(i_T, i_Z ... i_A, i_B) c! (t_T, i_Z, i_A, i_B) b! (t_T, i_Z, i_A, e_B)$]_2$) = Θ_2 (i_Z b! i_Z) = t!

CONICET

UNC

Are distributed schedulers what we need?



$\mathcal{I}\,(\,(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)\,) = 3$

$\mathcal{I}\,(\,(i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B)\,) = 4$

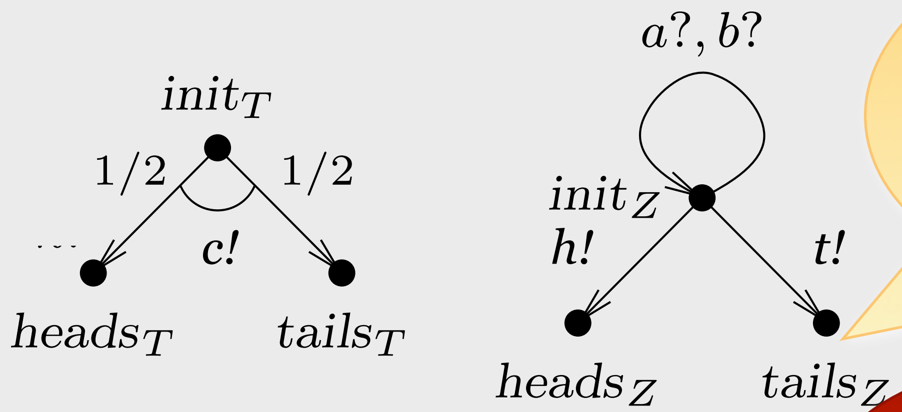
$[(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)]_3 \;=\; i_A \;=\; [(i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B)]_3$

$[(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)]_4 \;=\; i_B \;=\; [(i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B)]_4$

CONICET

UNC

Are distributed schedulers what we need?



None of components 3 and 4 can distinguish the system after these two executions

… and yet they are consider diferently

$$\mathcal{I}\left((i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B) \right) = 3$$

$$\mathcal{I}\left((i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B) \right) = 4$$

$$[(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)]_3 = i_A = [(i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B)]_3$$

$$[(i_T, i_Z, i_A, i_B)\ c!\ (h_T, i_Z, i_A, i_B)]_4 = i_B = [(i_T, i_Z, i_A, i_B)\ c!\ (t_T, i_Z, i_A, i_B)]_4$$

Strongly distributed schedulers

A strongly distributed scheduler is a distributed scheduler where I (the interleaving scheduler) meets the following condition:

for all $\sigma, \sigma' \in \mathsf{Frag}$ and components A_i, A_j,

such that $[\sigma]_i = [\sigma']_i, [\sigma]_j = [\sigma']_j$, it holds that

$$\frac{\mathcal{I}(\sigma)(i)}{\mathcal{I}(\sigma)(i) + \mathcal{I}(\sigma)(j)} = \frac{\mathcal{I}(\sigma')(i)}{\mathcal{I}(\sigma')(i) + \mathcal{I}(\sigma')(j)}$$

provided $\mathcal{I}(\sigma)(i) + \mathcal{I}(\sigma)(j) \neq 0 \neq \mathcal{I}(\sigma')(i) + \mathcal{I}(\sigma')(j)$

CONICET

UNC

Strongly distrib

If two components cannot distinguish two executions, their relative probabilities after such executions must be the same

A strongly distributed sched
scheduler where I (the interleavin eets
the following condition:

for all $\sigma, \sigma' \in \mathsf{Frag}$ and components A_i, A_j,

such that $[\sigma]_i = [\sigma']_i, [\sigma]_j = [\sigma']_j$, it holds that

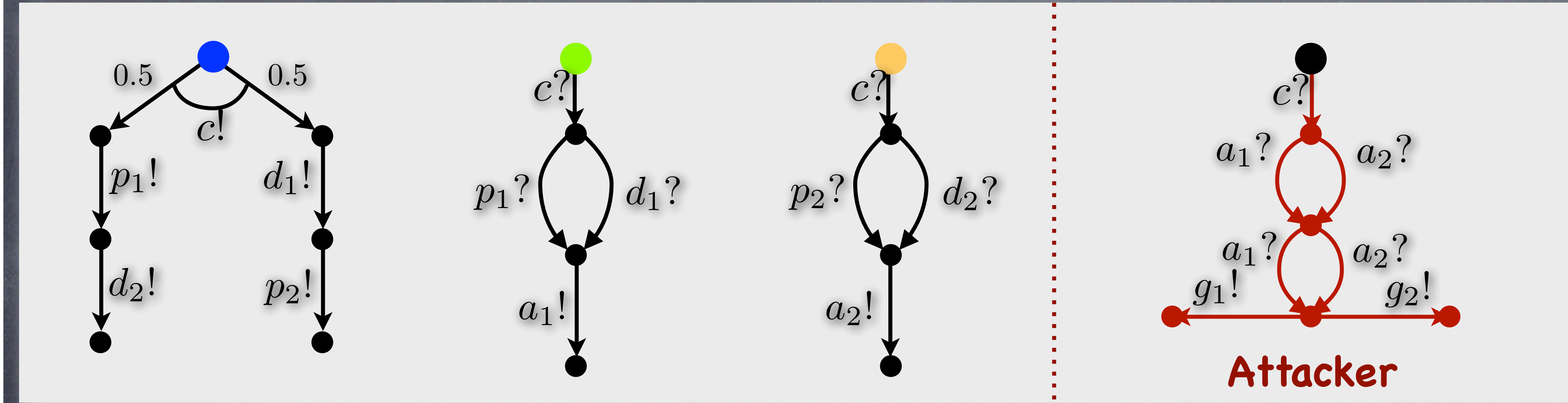
$$\frac{\mathcal{I}(\sigma)(i)}{\mathcal{I}(\sigma)(i) + \mathcal{I}(\sigma)(j)} = \frac{\mathcal{I}(\sigma')(i)}{\mathcal{I}(\sigma')(i) + \mathcal{I}(\sigma')(j)}$$

provided $\mathcal{I}(\sigma)(i) + \mathcal{I}(\sigma)(j) \neq 0 \neq \mathcal{I}(\sigma')(i) + \mathcal{I}(\sigma')(j)$

CONICET

UNC

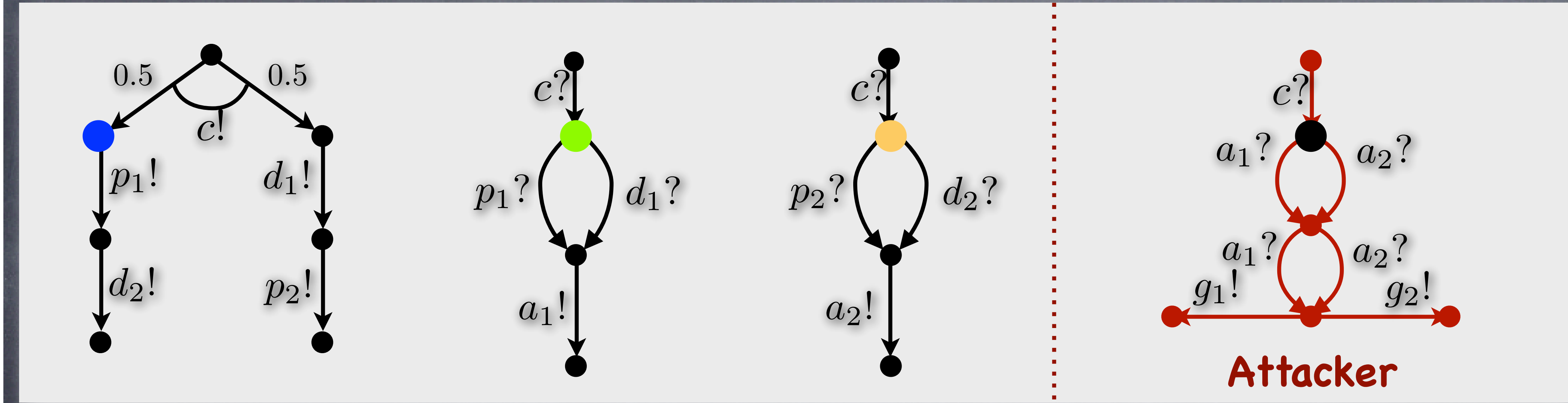
What about security?



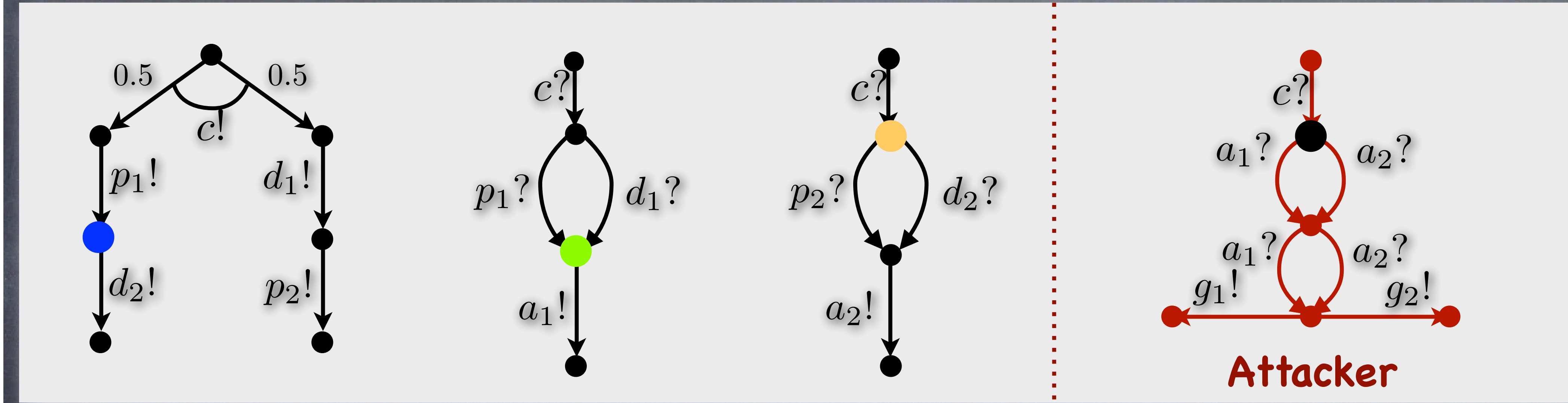
CONICET

UNC

What about security?



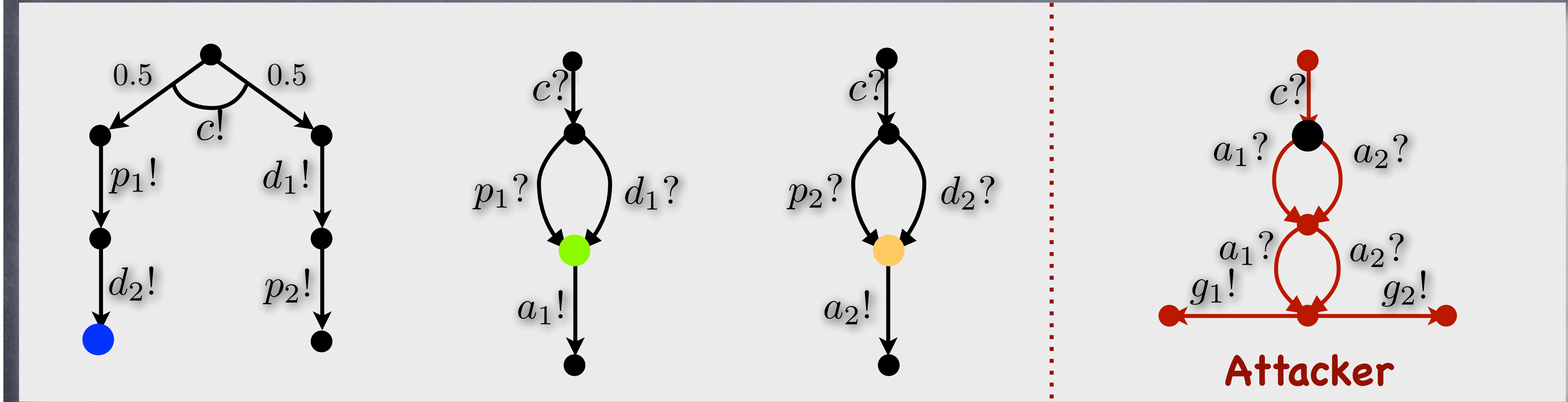
What about security?



CONICET

UNC

What about security?

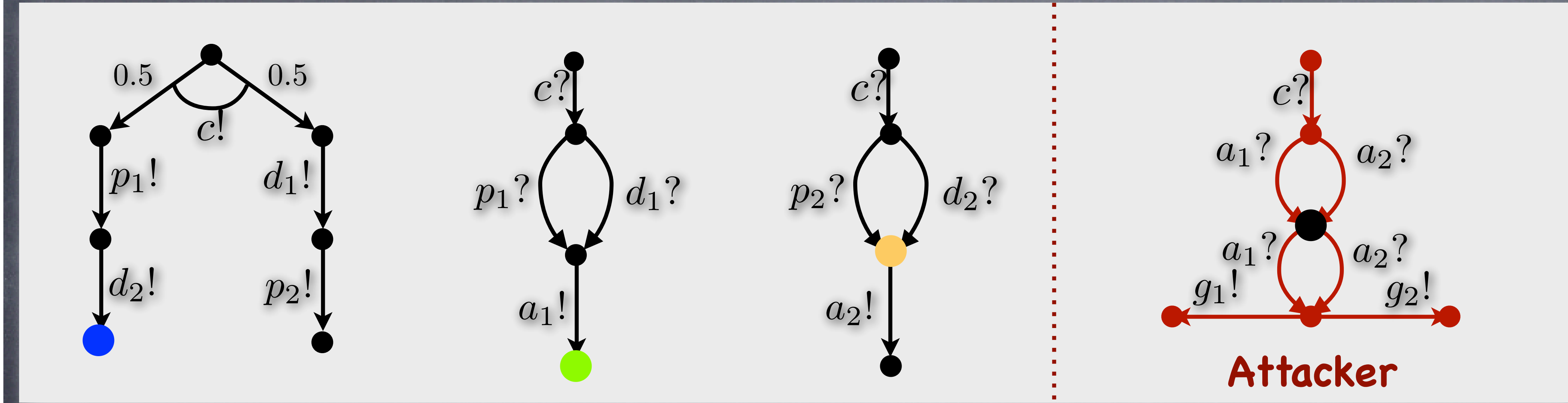


$\mathcal{I}(c\ p_1\ d_2) = 1$

CONICET

UNC

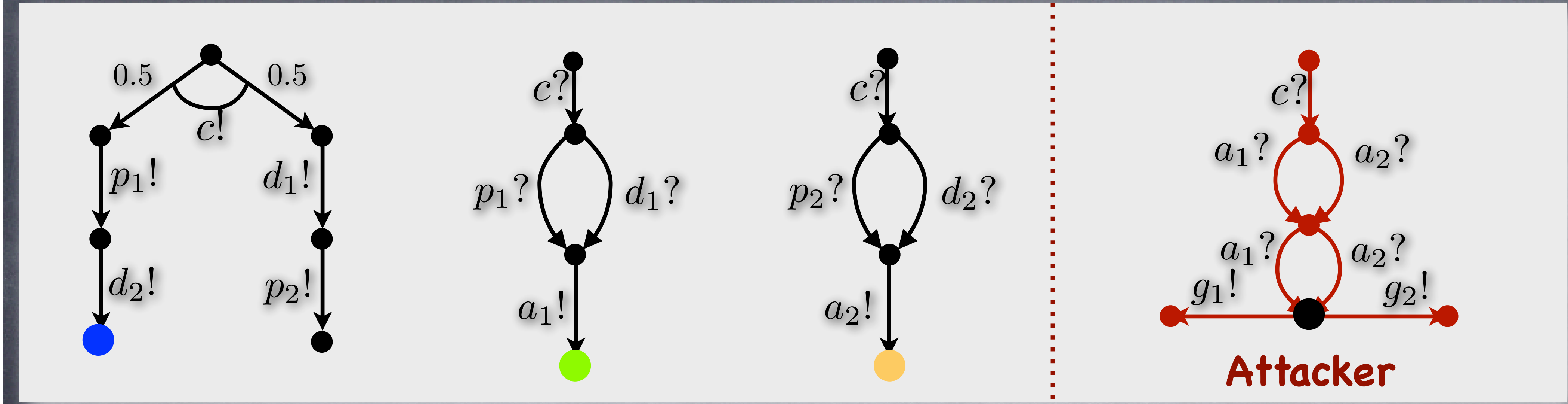
What about security?



$I(c\ p_1\ d_2) = 1$

$I(c\ p_1\ d_2\ a_1) = 2$

What about security?

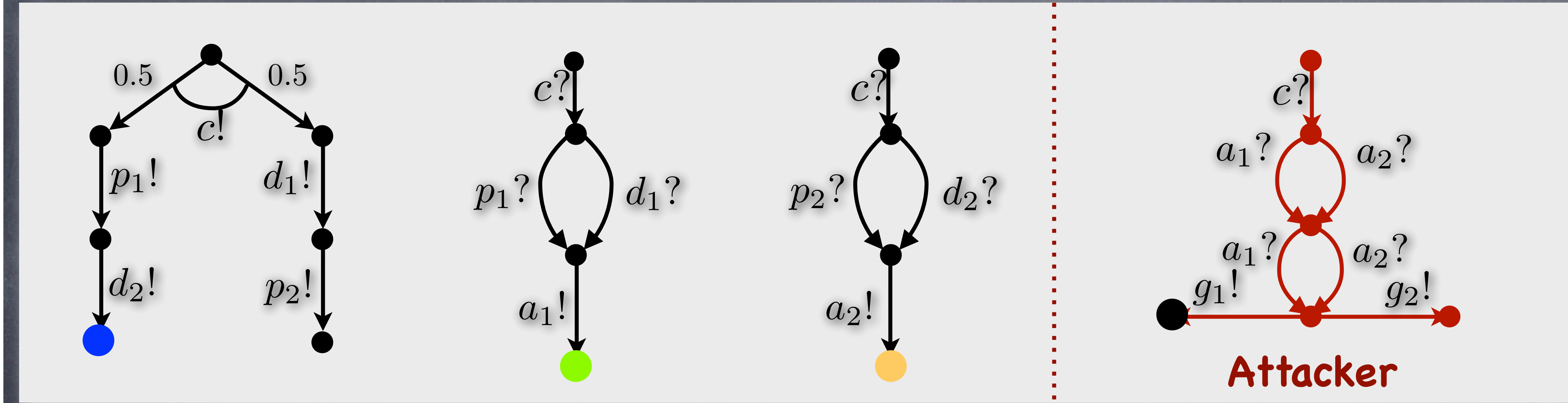


$I(c\ p_1\ d_2) = 1$

$I(c\ p_1\ d_2\ a_1) = 2$

$I(c\ p_1\ d_2\ a_1\ a_2) = \text{Atck}$

What about security?



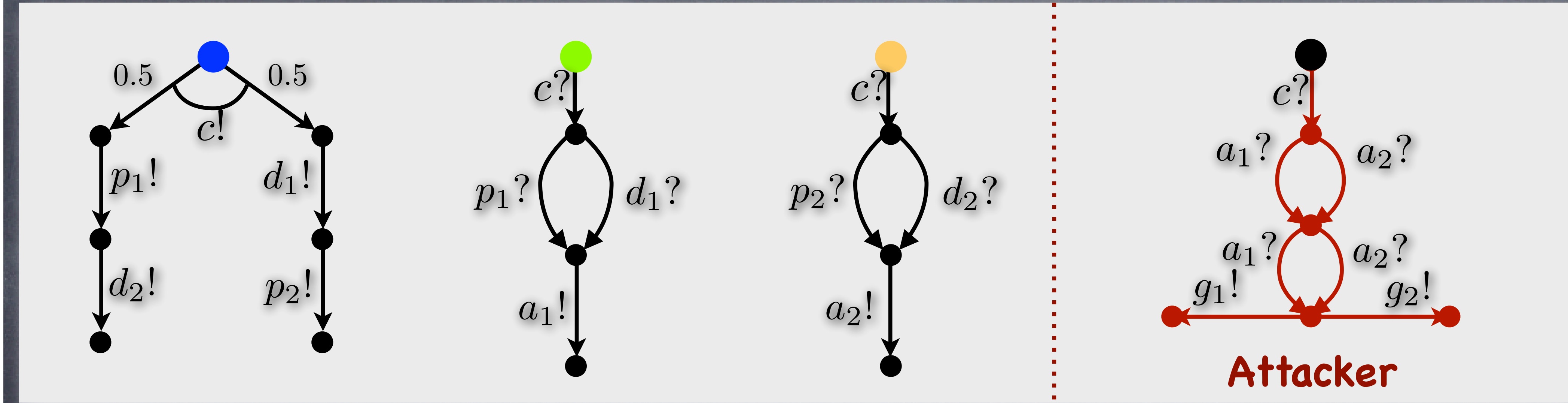
$\mathcal{I}(\, c \;\; p_1 \;\; d_2 \,) = 1$

$\mathcal{I}(\, c \;\; p_1 \;\; d_2 \;\; a_1 \,) = 2$

$\mathcal{I}(\, c \;\; p_1 \;\; d_2 \;\; a_1 \;\; a_2 \,) =$ Atck

Atacker guesses 1

What about security?



$I(c\ p_1\ d_2) = 1$

$I(c\ p_1\ d_2\ a_1) = 2$

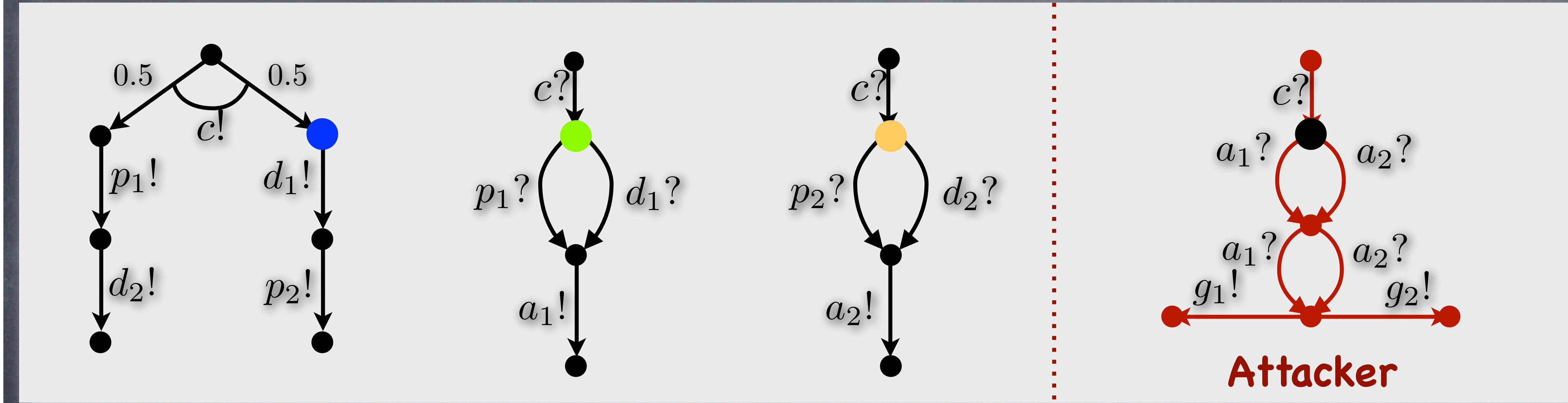
$I(c\ p_1\ d_2\ a_1\ a_2) =$ Atck

Atacker guesses 1

CONICET

UNC

What about security?



$\mathcal{I}(\ c\ \ p_1\ \ d_2\ ) = 1$

$\mathcal{I}(\ c\ \ p_1\ \ d_2\ \ a_1\ ) = 2$

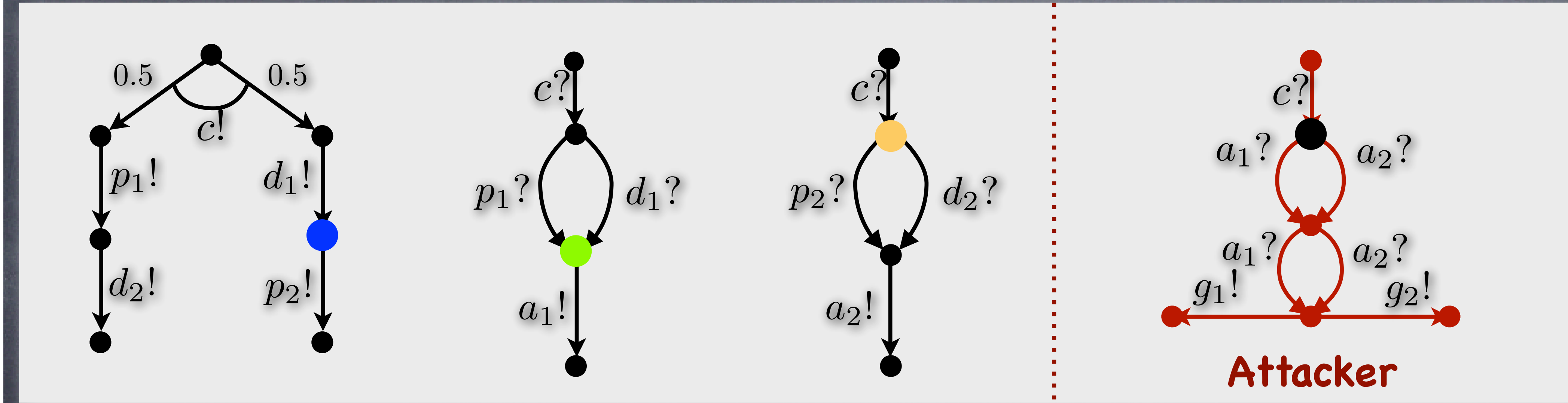
$\mathcal{I}(\ c\ \ p_1\ \ d_2\ \ a_1\ \ a_2\ ) =$ Atck

Atacker guesses 1

CONICET

UNC

What about security?



$I(c\ p_1\ d_2) = 1$

$I(c\ p_1\ d_2\ a_1) = 2$

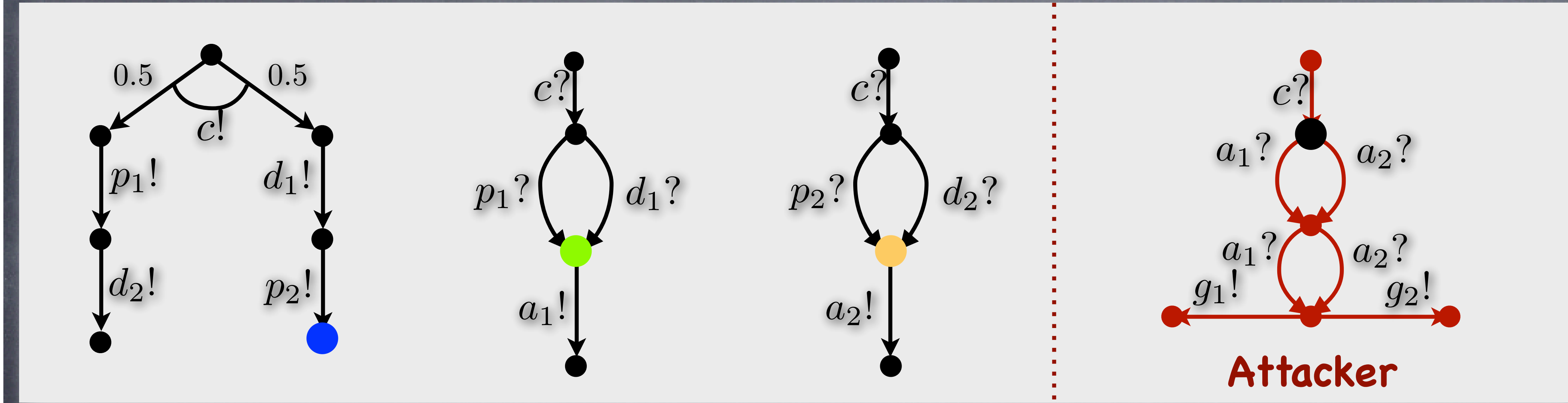
$I(c\ p_1\ d_2\ a_1\ a_2) = \text{Atck}$

Atacker guesses 1

CONICET

UNC

What about security?



$I(\underline{c\ p_1\ d_2}) = 1$

$I(c\ p_1\ d_2\ a_1) = 2$

$I(c\ p_1\ d_2\ a_1\ a_2) = \text{Atck}$

Atacker guesses 1

$I(\underline{c\ d_1\ p_2}) = 2$

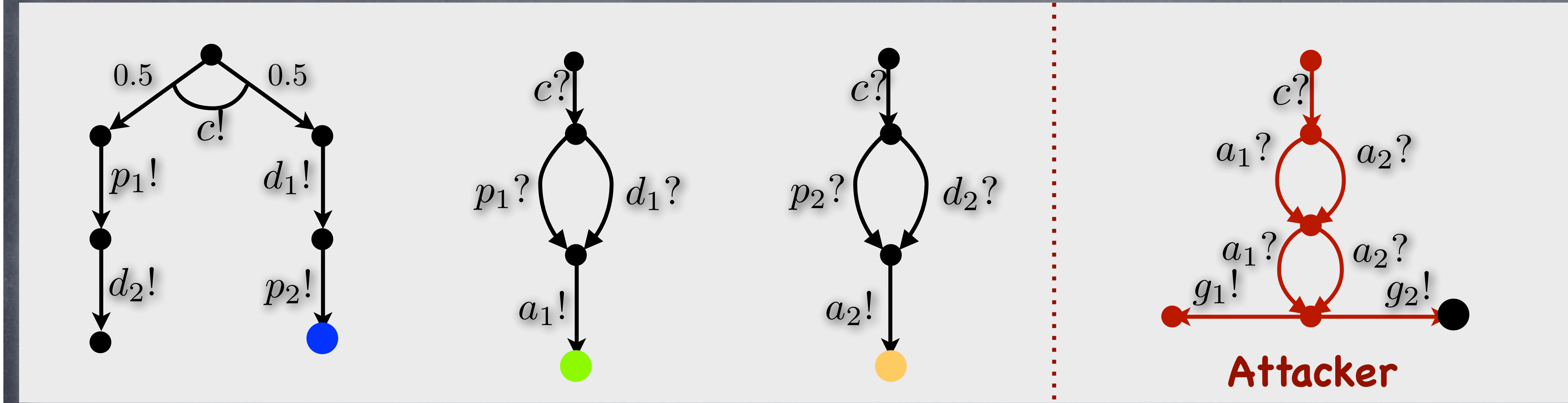
$$[\,c\ p_1\ d_2\,]_1 = p_1 \neq d_1 = [\,c\ d_1\ p_2\,]_1$$

$$[\,c\ p_1\ d_2\,]_2 = p_2 \neq d_2 = [\,c\ d_1\ p_2\,]_2$$

CONICET

UNC

What about security?



$I(\underline{c\ p_1\ d_2}) = 1$

$I(c\ p_1\ d_2\ a_1) = 2$

$I(c\ p_1\ d_2\ \underline{a_1\ a_2}) =$ Atck

Atacker guesses 1

$I(\underline{c\ d_1\ p_2}) = 2$

$I(c\ d_1\ p_2\ a_2) = 1$

$I(c\ d_1\ p_2\ \underline{a_2\ a_1}) =$ Atck

Atacker guesses 2 ☹

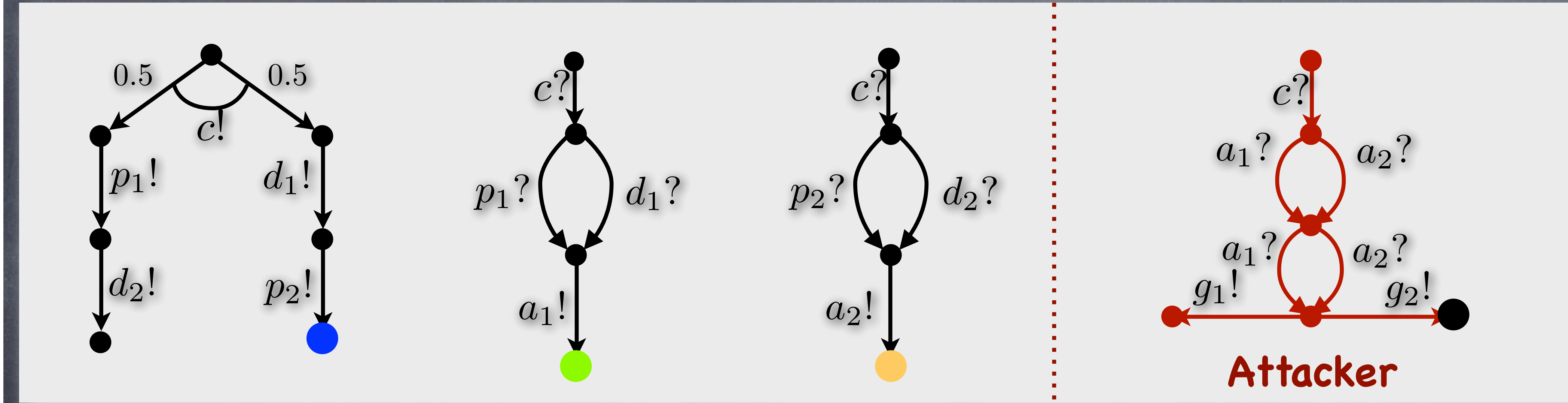
$$[\,c\ p_1\ d_2\,]_1 = p_1 \neq d_1 = [\,c\ d_1\ p_2\,]_1$$

$$[\,c\ p_1\ d_2\,]_2 = p_2 \neq d_2 = [\,c\ d_1\ p_2\,]_2$$

CONICET

UNC

What about security?



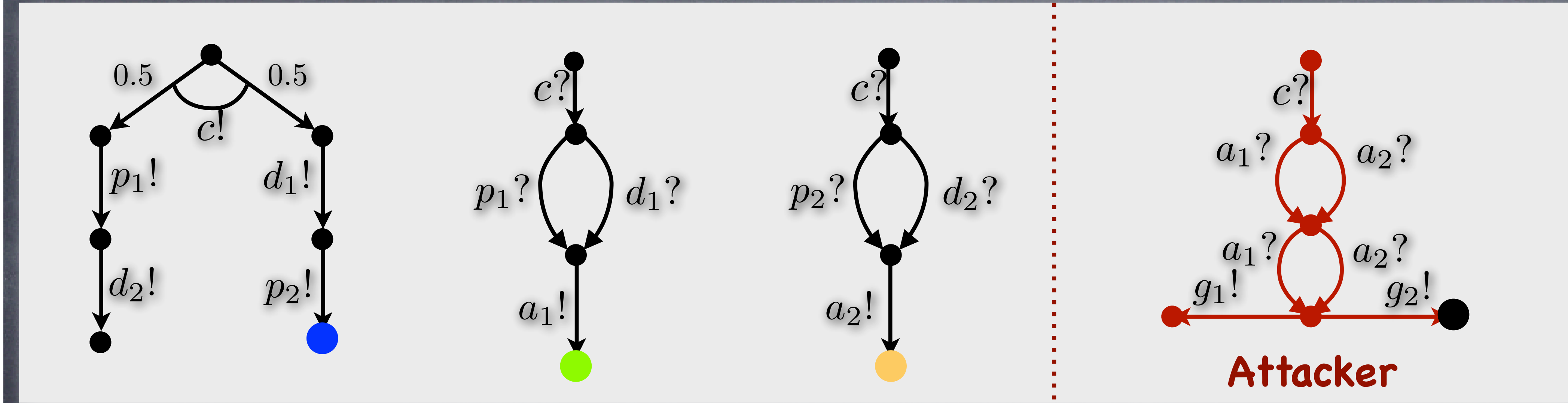
$$[\ c\ \ p_1\ \ d_2\ ]_1 = p_1 \neq d_1 = [\ c\ \ d_1\ \ p_2\ ]_1$$

$$[\ c\ \ p_1\ \ d_2\ ]_2 = p_2 \neq d_2 = [\ c\ \ d_1\ \ p_2\ ]_2$$

CONICET

UNC

What about security?



$$[\ c\ \ p_1\ \ d_2\ ]_1 = p_1 \approx d_1 = [\ c\ \ d_1\ \ p_2\ ]_1$$

$$[\ c\ \ p_1\ \ d_2\ ]_2 = p_2 \approx d_2 = [\ c\ \ d_1\ \ p_2\ ]_2$$

secret actions should not be distinguished by I

CONICET

UNC

Distributed schedulers under secrecy

A distributed scheduler under secrecy is a distributed scheduler where I meets the following condition:

for all $\sigma, \sigma' \in \mathsf{Frag}$ and components A_i, A_j,

such that $[\sigma]_i \approx [\sigma']_i, [\sigma]_j \approx [\sigma']_j$, it holds that

$$\frac{\mathcal{I}(\sigma)(i)}{\mathcal{I}(\sigma)(i) + \mathcal{I}(\sigma)(j)} = \frac{\mathcal{I}(\sigma')(i)}{\mathcal{I}(\sigma')(i) + \mathcal{I}(\sigma')(j)}$$

provided $\mathcal{I}(\sigma)(i) + \mathcal{I}(\sigma)(j) \neq 0 \neq \mathcal{I}(\sigma')(i) + \mathcal{I}(\sigma')(j)$

$$(\forall a \in O_i : \mathsf{last}([\sigma]_i) \xrightarrow{a}_i) \text{ iff } (\forall a \in O_i : \mathsf{last}([\sigma']_i) \xrightarrow{a}_i)$$

$$(\forall a \in O_j : \mathsf{last}([\sigma]_j) \xrightarrow{a}_j) \text{ iff } (\forall a \in O_j : \mathsf{last}([\sigma']_j) \xrightarrow{a}_j)$$

CONICET

UNC

[Pelozo & D'Argenio 2012]

Results (finite state models)

	Dist. Sched.	Str. Dist. Sched.	Distr. Sched. with Secrecy
Det = Random sup P(F goal)	Yes	No	No
sup P(F goal)?	Undecidable	Undecidable	Undecidable
sup P(F goal) = 1	Undecidable	Undecidable	Undecidable

sup P(F goal)? is NP-Hard for

(locally | globally) memoryless (deterministic | randomized)

distributed schedulers

Results (finite state systems)

- Partial order reduction:
  - Peled's original conditions preserve strongly distributed schedulers
  - Apply classical algorithms for prob. MC on reduction
- Counterexample guided refinement:
  - Check sup P(F goal) ≤ p with classical Prob. MC
  - If the result is true => the model sat. property
  - If not and counterexample is a DS => error
  - If not and counterexample is not a DS

    => refine model and recalculate

CONICET

UNC

Results (finite behaviour systems)

- Bounded reachability reduces to a polynomial optimization problem
  - For dist. schedulers variables take value in {0,1}
  - For SDS/DSS => quadratic restrictions
- Anonymity can be encoded on SMTs
  - through a system of polynomial inequalities
- A good thing:
  - Usually, security protocols are of finite behavior
- A bad thing:
  - Inherently exponential

CONICET

UNC

Conclusion

Ignored by classical probabilistic model checking

- Distributed schedulers properly captures the idea of partial observation among components.
- Particularly suited for security.
- These observations has been acknowledge by other authors from different point of views:
  - Compositionality [de Alfaro, Henzinger, Jhala, 2001], [Cheung, Lynch, Segala, Vaandrager, 2006]
  - Security analysis [Chatzikokolakis, Palamidessi, 2007], [Andrés, Palamidessi, Rossum, Sokolova, 2011], [Chadha Sistla, Viswanathan, 2010]
  - Testing [Georgievska, Andova, 2010], [Hierons, Núñez 2012]
- Down side: undecidability and complexity

CONICET

UNC

Conclusion

Ignored by classical probabilistic model checking

- Distributed schedulers properly captures the idea of partial observation among components.
- Par... for security.

e.g.:tractable but interesting subclasses / abstraction techniques / appropriate data structures / etc.

...en acknowledge by other ...nt of views:

- ... [de Alfaro, Henzi... ...ch, Segala, Vaandrager, ...06]
- Security analys... [Chatz... Palamidessi, Rossum, Sokolova, 2011], ...
- Testing [Georgievska, Andova ...10], [Hierons ...nez 2012]

They did not stop the automated theorem proving or SAT solving communities (among others)

- Down side: undecidability and complexity

CONICET

UNC

Analysis of Distributed Probabilistic Systems: Limitations and Possibilities

Pedro R. D'Argenio

Universidad Nacional de Córdoba
CONICET

Joint work with Sergio Giro, Luis M. Ferrer Fioriti, Georgel Calin, Pepijn Crouzen, Ernst Moritz Hahn, Lijun Zhang, Silvia Pelozo

http://dsg.cs.famaf.unc.edu.ar/

CONICET

19-Jun-2014 - OPCT - Bertinoro

UNC

References:

S. Giro, P.R. D'Argenio: On the Expressive Power of Schedulers in Distributed Probabilistic Systems. QAPL 2009 (ENTCS 253(3):45-71).

S. Giro, P.R. D'Argenio: Quantitative Model Checking Revisited: Neither Decidable Nor Approximable. FORMATS 2007: 179-194.

S. Giro, P.R. D'Argenio, L.M. Ferrer Fioriti: Partial Order Reduction for Probabilistic Systems: A Revision for Distributed Schedulers. CONCUR 2009: 338-353.

S. Giro: On the Automatic Verification of Distributed Probabilistic Automata with Partial Information. PhD thesis, FaMAF, UNC, 2010.

L.M. Ferrer Fioriti: Reducción de orden parcial en model checking probabilista simbólico. Lic. thesis, FaMAF, UNC, 2010.

G. Calin, P. Crouzen, P.R. D'Argenio, E.M. Hahn, L. Zhang: Time-Bounded Reachability in Distributed Input/Output Interactive Probabilistic Chains. SPIN 2010: 193-211.

S. Pelozo, P.R. D'Argenio: Security analysis in probabilistic distributed protocols via bounded reachability. TGC 2012: 182-197.

S. Giro, M.N. Rabe: Verification of Partial-Information Probabilistic Systems Using Counterexample-Guided Refinements. ATVA 2012: 333-348.

S. Giro, P.R. D'Argenio, and L.M. Ferrer Fioriti: Distributed probabilistic input/output automata: Expressiveness, (un)decidability and algorithms. TCS 538:84-102, 2014.

CONICET

UNC